

EL792821573

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR LETTERS PATENT

**Application Program Interface for Network Software
Platform**

Inventor(s):
Brad Abrams

ATTORNEY'S DOCKET NO. MS1-862US

TECHNICAL FIELD

This invention relates to network software, such as Web applications, and to computer software development of such network software. More particularly, this invention relates to an application program interface (API) that facilitates use of a network software platform by application programs and computer hardware.

BACKGROUND

Very early on, computer software came to be categorized as “operating system” software or “application” software. Broadly speaking, an application is software meant to perform a specific task for the computer user such as solving a mathematical equation or supporting word processing. The operating system is the software that manages and controls the computer hardware. The goal of the operating system is to make the computer resources available to the application programmer while at the same time, hiding the complexity necessary to actually control the hardware.

The operating system makes the resources available via functions that are collectively known as the Application Program Interface or API. The term API is also used in reference to a single one of these functions. The functions are often grouped in terms of what resource or service they provide to the application programmer. Application software requests resources by calling individual API functions. API functions also serve as the means by which messages and information provided by the operating system are relayed back to the application software.

In addition to changes in hardware, another factor driving the evolution of operating system software has been the desire to simplify and speed application

1 software development. Application software development can be a daunting task,
2 sometimes requiring years of developer time to create a sophisticated program
3 with millions of lines of code. For a popular operating system such as Microsoft
4 Windows®, application software developers write thousands of different
5 applications each year that utilize the operating system. A coherent and usable
6 operating system base is required to support so many diverse application
7 developers.

8 Often, development of application software can be made simpler by making
9 the operating system more complex. That is, if a function may be useful to several
10 different application programs, it may be better to write it once for inclusion in the
11 operating system, than requiring dozens of software developers to write it dozens
12 of times for inclusion in dozens of different applications. In this manner, if the
13 operating system supports a wide range of common functionality required by a
14 number of applications, significant savings in applications software development
15 costs and time can be achieved.

16 Regardless of where the line between operating system and application
17 software is drawn, it is clear that for a useful operating system, the API between
18 the operating system and the computer hardware and application software is as
19 important as efficient internal operation of the operating system itself.

20 Over the past few years, the universal adoption of the Internet, and
21 networking technology in general, has changed the landscape for computer
22 software developers. Traditionally, software developers focused on single-site
23 software applications for standalone desktop computers, or LAN-based computers
24 that were connected to a limited number of other computers via a local area
25 network (LAN). Such software applications were typically referred to as “shrink

1 wrapped” products because the software was marketed and sold in a shrink-
2 wrapped package. The applications utilized well-defined APIs to access the
3 underlying operating system of the computer.

4 As the Internet evolved and gained widespread acceptance, the industry
5 began to recognize the power of hosting applications at various sites on the World
6 Wide Web (or simply the “Web”). In the networked world, clients from anywhere
7 could submit requests to server-based applications hosted at diverse locations and
8 receive responses back in fractions of a second. These Web applications, however,
9 were typically developed using the same operating system platform that was
10 originally developed for standalone computing machines or locally networked
11 computers. Unfortunately, in some instances, these applications do not adequately
12 transfer to the distributed computing regime. The underlying platform was simply
13 not constructed with the idea of supporting limitless numbers of interconnected
14 computers.

15 To accommodate the shift to the distributed computing environment being
16 ushered in by the Internet, Microsoft Corporation is developing a network
17 software platform known as the “.NET” platform (read as “Dot Net”). The
18 platform allows developers to create Web services that will execute over the
19 Internet. Such a dynamic shift requires a new ground-up design of an entirely new
20 API.

21 In response to this challenge, the inventors developed a unique set of API
22 functions for Microsoft’s .NET™ platform.
23
24
25

SUMMARY

An application program interface (API) provides a set of functions, including a set of base classes and types that are used in substantially all applications accessing the API, for application developers who build Web applications on a network platform, such as Microsoft Corporation's .NET™ platform.

BRIEF DESCRIPTION OF THE DRAWINGS

The same numbers are used throughout the drawings to reference like features.

Fig. 1 illustrates a network architecture in which clients access Web services over the Internet using conventional protocols.

Fig. 2 is a block diagram of a software architecture for Microsoft's .NET™ platform, which includes an application program interface (API).

Fig. 3 is a block diagram of unique namespaces supported by the API, as well as function classes of the various API functions.

Fig. 4 is a block diagram of an exemplary computer that may execute all or part of the software architecture.

BRIEF DESCRIPTION OF ACCOMPANYING COMPACT DISC

Accompanying this specification is a compact disc that stores a compiled HTML help file identifying the API (application program interface) for Microsoft's .NET™ network platform. The file is named "cpref.chm" and was created on June 8, 2001. It is 30.81 Mbytes in size. The file can be executed on a Windows®-based computing device (e.g., IBM-PC, or equivalent) that executes a

1 Windows®-brand operating system (e.g., Windows® NT, Windows® 98,
2 Windows® 2000, etc.). The compiled HTML help file stored on the compact disk
3 is hereby incorporated by reference.

4 Additionally, the APIs contained in the compiled HTML help file are also
5 provided in approximately 100 separate text files named "NamespaceName.txt".
6 The text files comply with the ASCII format.

7 The compact disc itself is a CD-ROM, and conforms to the ISO 9660
8 standard.

9 10 **DETAILED DESCRIPTION**

11 This disclosure addresses an application program interface (API) for a
12 network platform upon which developers can build Web applications and services.
13 More particularly, an exemplary API is described for the .NET™ platform created
14 by Microsoft Corporation. The .NET™ platform is a software platform for Web
15 services and Web applications implemented in the distributed computing
16 environment. It represents the next generation of Internet computing, using open
17 communication standards to communicate among loosely coupled Web services
18 that are collaborating to perform a particular task.

19 In the described implementation, the .NET™ platform utilizes XML
20 (extensible markup language), an open standard for describing data. XML is
21 managed by the World Wide Web Consortium (W3C). XML is used for defining
22 data elements on a Web page and business-to-business documents. XML uses a
23 similar tag structure as HTML; however, whereas HTML defines how elements
24 are displayed, XML defines what those elements contain. HTML uses predefined
25 tags, but XML allows tags to be defined by the developer of the page. Thus,

1 virtually any data items can be identified, allowing Web pages to function like
2 database records. Through the use of XML and other open protocols, such as
3 Simple Object Access Protocol (SOAP), the .NET™ platform allows integration of
4 a wide range of services that can be tailored to the needs of the user. Although the
5 embodiments described herein are described in conjunction with XML and other
6 open standards, such are not required for the operation of the claimed invention.
7 Other equally viable technologies will suffice to implement the inventions
8 described herein.

9 10 EXEMPLARY NETWORK ENVIRONMENT

11 Fig. 1 shows a network environment 100 in which a network platform, such
12 as the .NET™ platform, may be implemented. The network environment 100
13 includes representative Web services 102(1), ..., 102(N), which provide services
14 that can be accessed over a network 104 (e.g., Internet). The Web services,
15 referenced generally as number 102, are programmable application components
16 that are reusable and interact programmatically over the network 104, typically
17 through industry standard Web protocols, such as XML, SOAP, WAP (wireless
18 application protocol), HTTP (hypertext transport protocol), and SMTP (simple
19 mail transfer protocol) although other means of interacting with the Web services
20 over the network may also be used, such as Remote Procedure Call (RPC) or
21 object broker type technology. A Web service can be self-describing and is often
22 defined in terms of formats and ordering of messages.

23 Web services 102 are accessible directly by other services (as represented
24 by communication link 106) or a software application, such as Web application
25 110 (as represented by communication links 112 and 114). Each Web service 102

1 is illustrated as including one or more servers that execute software to handle
2 requests for particular services. Such services often maintain databases that store
3 information to be served back to requesters. Web services may be configured to
4 perform any one of a variety of different services. Examples of Web services
5 include login verification, notification, database storage, stock quoting, location
6 directories, mapping, music, electronic wallet, calendar/scheduler, telephone
7 listings, news and information, games, ticketing, and so on. The Web services can
8 be combined with each other and with other applications to build intelligent
9 interactive experiences.

10 The network environment 100 also includes representative client devices
11 120(1), 120(2), 120(3), 120(4), ..., 120(M) that utilize the Web services 102 (as
12 represented by communication link 122) and/or the Web application 110 (as
13 represented by communication links 124, 126, and 128). The clients may
14 communicate with one another using standard protocols as well, as represented by
15 an exemplary XML link 130 between clients 120(3) and 120(4).

16 The client devices, referenced generally as number 120, can be
17 implemented many different ways. Examples of possible client implementations
18 include, without limitation, portable computers, stationary computers, tablet PCs,
19 televisions/set-top boxes, wireless communication devices, personal digital
20 assistants, gaming consoles, printers, photocopiers, and other smart devices.

21 The Web application 110 is an application designed to run on the network
22 platform and may utilize the Web services 102 when handling and servicing
23 requests from clients 120. The Web application 110 is composed of one or more
24 software applications 130 that run atop a programming framework 132, which are
25 executing on one or more servers 134 or other computer systems. Note that a

1 portion of Web application 110 may actually reside on one or more of clients 120.
2 Alternatively, Web application 110 may coordinate with other software on clients
3 120 to actually accomplish its tasks.

4 The programming framework 132 is the structure that supports the
5 applications and services developed by application developers. It permits multi-
6 language development and seamless integration by supporting multiple languages.
7 It supports open protocols, such as SOAP, and encapsulates the underlying
8 operating system and object model services. The framework provides a robust and
9 secure execution environment for the multiple programming languages and offers
10 secure, integrated class libraries.

11 The framework 132 is a multi-tiered architecture that includes an
12 application program interface (API) layer 142, a common language runtime (CLR)
13 layer 144, and an operating system/services layer 146. This layered architecture
14 allows updates and modifications to various layers without impacting other
15 portions of the framework. A common language specification (CLS) 140 allows
16 designers of various languages to write code that is able to access underlying
17 library functionality. The specification 140 functions as a contract between
18 language designers and library designers. By adhering to the CLS, libraries
19 written in one language can be directly accessible to code modules written in other
20 languages to achieve seamless integration between code modules written in one
21 language and code modules written in another language.

22 The API layer 142 presents groups of functions that the applications 130
23 can call to access the resources and services provided by layer 146. By exposing
24 the API functions for a network platform, application developers can create Web
25 applications for distributed computing systems that make full use of the network

resources and other Web services, without needing to understand the complex interworkings of how those network resources actually operate or are made available. Moreover, the Web applications can be written in any number of programming languages, and translated into an intermediate language supported by the common language runtime 144 and included as part of the common language specification 140. . In this way, the API layer 142 can provide methods for a wide and diverse variety of applications.

Additionally, the framework 132 can be configured to support API calls placed by remote applications executing remotely from the servers 134 that host the framework. Representative applications 148(1) and 148(2) residing on clients 120(3) and 120(M), respectively, can use the API functions by making calls directly, or indirectly, to the API layer 142 over the network 104.

The framework may also be implemented at the clients. Client 120(3) represents the situation where a framework 150 is implemented at the client. This framework may be identical to server-based framework 132, or modified for client purposes. Alternatively, the client-based framework may be condensed in the event that the client is a limited or dedicated function device, such as a cellular phone, personal digital assistant, handheld computer, or other communication/computing device.

DEVELOPERS' PROGRAMMING FRAMEWORK

Fig. 2 shows the programming framework 132 in more detail. The common language specification (CLS) layer 140 supports applications written in a variety of languages 130(1), 130(2), 130(3), 130(4), ..., 130(K). Such application languages include Visual Basic, C++, C#, COBOL, Jscript, Perl, Eiffel, Python,

1 and so on. The common language specification 140 specifies a subset of features
2 or rules about features that, if followed, allow the various languages to
3 communicate. For example, some languages do not support a given type (e.g., an
4 "int*" type) that might otherwise be supported by the common language runtime
5 144. In this case, the common language specification 140 does not include the
6 type. On the other hand, types that are supported by all or most languages (e.g.,
7 the "int[]" type) is included in common language specification 140 so library
8 developers are free to use it and are assured that the languages can handle it. This
9 ability to communicate results in seamless integration between code modules
10 written in one language and code modules written in another language. Since
11 different languages are particularly well suited to particular tasks, the seamless
12 integration between languages allows a developer to select a particular language
13 for a particular code module with the ability to use that code module with modules
14 written in different languages. The common language runtime 144 allow seamless
15 multi-language development, with cross language inheritance, and provide a
16 robust and secure execution environment for the multiple programming languages.
17 For more information on the common language specification 140 and the common
18 language runtime 144, the reader is directed to co-pending applications entitled
19 "Method and System for Compiling Multiple Languages", filed 6/21/2000 (serial
20 number 09/598,105) and "Unified Data Type System and Method" filed 7/10/2000
21 (serial number 09/613,289), which are incorporated by reference.

22 The framework 132 encapsulates the operating system 146(1) (e.g.,
23 Windows®-brand operating systems) and object model services 146(2) (e.g.,
24 Component Object Model (COM) or Distributed COM). The operating system
25 146(1) provides conventional functions, such as file management, notification,

1 event handling, user interfaces (e.g., windowing, menus, dialogs, etc.), security,
2 authentication, verification, processes and threads, memory management, and so
3 on. The object model services 146(2) provide interfacing with other objects to
4 perform various tasks. Calls made to the API layer 142 are handed to the common
5 language runtime layer 144 for local execution by the operating system 146(1)
6 and/or object model services 146(2).

7 The API 142 groups API functions into multiple namespaces. Namespaces
8 essentially define a collection of classes, interfaces, delegates, enumerations, and
9 structures, which are collectively called “types”, that provide a specific set of
10 related functionality. A class represents managed heap allocated data that has
11 reference assignment semantics. A delegate is an object oriented function pointer.
12 An enumeration is a special kind of value type that represents named constants. A
13 structure represents static allocated data that has value assignment semantics. An
14 interface defines a contract that other types can implement.

15 By using namespaces, a designer can organize a set of types into a
16 hierarchical namespace. The designer is able to create multiple groups from the
17 set of types, with each group containing at least one type that exposes logically
18 related functionality. In the exemplary implementation, the API 142 is organized
19 into four root namespaces: a first namespace 200 for Web applications, a second
20 namespace 202 for client applications, a third namespace 204 for data and XML,
21 and a fourth namespace 206 for base class libraries (BCLs). Each group can then
22 be assigned a name. For instance, types in the Web applications namespace 200
23 are assigned the name “Web”, and types in the data and XML namespace 204 can
24 be assigned names “Data” and “XML” respectively. The named groups can be
25 organized under a single “global root” namespace for system level APIs, such as

an overall System namespace. By selecting and prefixing a top level identifier, the types in each group can be easily referenced by a hierarchical name that includes the selected top level identifier prefixed to the name of the group containing the type. For instance, types in the Web applications namespace 200 can be referenced using the hierarchical name "System.Web". In this way, the individual namespaces 200, 202, 204, and 206 become major branches off of the System namespace and can carry a designation where the individual namespaces are prefixed with a designator, such as a "System." prefix.

The Web applications namespace 200 pertains to Web based functionality, such as dynamically generated Web pages (e.g., Microsoft's Active Server Pages (ASP)). It supplies types that enable browser/server communication. The client applications namespace 202 pertains to drawing and client side UI functionality. It supplies types that enable drawing of two-dimensional (2D) and three-dimensional (3D) drawings, imaging, and printing, as well as the ability to construct window forms, menus, boxes, and so on.

The data and XML namespace 204 relates to connectivity to data sources and XML functionality. It supplies classes, interfaces, delegates, and enumerations that enable security, specify data types, and serialize objects into XML format documents or streams. The base class libraries (BCL) namespace 206 pertains to basic system and runtime functionality. It contains the fundamental types and base classes that define commonly-used value and reference data types, events and event handlers, interfaces, attributes, and processing exceptions.

In addition to the framework 132, programming tools 210 are provided to assist the developer in building Web services and/or applications. One example of

1 the programming tools 200 is Visual Studio™, a multi-language suite of
2 programming tools offered by Microsoft Corporation.

4 ROOT API NAMESPACES

5 Fig. 3 shows the API 142 and its four root namespaces in more detail. In
6 one embodiment, the namespaces are identified according to a hierarchical naming
7 convention in which strings of names are concatenated with periods. For instance,
8 the Web applications namespace 200 is identified by the root name
9 “System.Web”. Within the “System.Web” namespace is another namespace for
10 Web services, identified as “System.Web.Services”, which further identifies
11 another namespace for a description known as
12 “System.Web.Services.Description”. With this naming convention in mind, the
13 following provides a general overview of selected namespaces of the API 142,
14 although other naming conventions could be used with equal effect.

15 The Web applications namespace 200 (“System.Web”) defines additional
16 namespaces, including:

- 18 • A services namespace 300 (“System.Web.Services”) containing
19 classes that enable a developer to build and use Web services. The
20 services namespace 300 defines additional namespaces, including a
21 description namespace 302 (“System.Web.Services.Description”) containing
22 classes that enable a developer to publicly describe a
23 Web service via a service description language (such as WSDL, a
24 specification available from the W3C), a discovery namespace 304
25 (“System.Web.Services.Discovery”) containing classes that allow

1 Web service consumers to locate available Web Services on a Web
2 server, and a protocols namespace 306
3 (“System.Web.Services.Protocols”) containing classes that define
4 the protocols used to transmit data across a network during
5 communication between Web service clients and the Web service
6 itself.

- 7 • A caching namespace 308 (“System.Web.Caching”) containing
8 classes that enable developers to decrease Web application response
9 time through temporarily caching frequently used resources on the
10 server. This includes ASP.NET pages, web services, and user
11 controls. (ASP.NET is the updated version of Microsoft’s ASP
12 technology.) Additionally, a cache dictionary is available for
13 developers to store frequently used resources, such as hash tables
14 and other data structures.
- 15 • A configuration namespace 310 (“System.Web.Configuration”) containing
16 classes that are used to read configuration data in for an
17 application.
- 18 • A UI namespace 312 (“System.Web.UI”) containing types that allow
19 developers to create controls and pages that will appear in Web
20 applications as user interfaces on a Web page. This namespace
21 includes the control class, which provides all web based controls,
22 whether those encapsulating HTML elements, higher level Web
23 controls, or even custom User controls, with a common set of
24 functionality. Also provided are classes which provide the web
25 forms server controls data binding functionality, the ability to save

the view state of a given control or page, as well as parsing functionality for both programmable and literal controls. Within the UI namespace 312 are two additional namespaces: an HTML controls namespace 314 (“System.Web.UI.HtmlControls”) containing classes that permit developers to interact with types that encapsulates html 3.2 elements create HTML controls, and a Web controls namespace 316 (“System.Web.UI.WebControls”) containing classes that allow developers to create higher level Web controls.

- A security namespace 318 (“System.Web.Security”) containing classes used to implement security in web server applications, such as basic authentication, challenge response authentication, and role based authentication.
- A session state namespace 320 (“System.Web.SessionState”) containing classes used to access session state values (i.e., data that lives across requests for the lifetime of the session) as well as session-level settings and lifetime management methods.

The client applications namespace 202 is composed of two namespaces:

- A windows forms namespace 322 (“System.Windows.Forms”) containing classes for creating Windows®-based client applications that take full advantage of the rich user interface features available in the Microsoft Windows® operating system, such as the ability to drag and drop screen elements. Such classes may include wrapped

1 APIs available in the Microsoft Windows® operating system that
2 are used in a windowing UI environment. Within this namespace
3 are a design namespace 324 (“System.Windows.Forms.Design”) that
4 contains classes to extend design-time support for Windows forms
5 and a component model namespace 326
6 (“System.Windows.Forms.ComponentModel”) that contains the
7 windows form implementation of the general component model
8 defined in System.ComponentModel. This namespace contains
9 designer tools, such as Visual Studio, which offer a rich experience
10 for developers at design time.

- 11 • A drawing namespace 328 (“System.Drawing”) containing classes
12 for graphics functionality. The drawing namespace 328 includes a
13 2D drawing namespace 330 (“System.Drawing.Drawing2D”) that
14 contains classes and enumerations to provide advanced 2-
15 dimensional and vector graphics functionality, an imaging
16 namespace 332 (“System.Drawing.Imaging”) that contains classes
17 for advanced imaging functionality, a printing namespace 334
18 (“System.Drawing.Printing”) that contains classes to permit
19 developers to customize printing, and a text namespace 336
20 (“System.Drawing.Text”) that contains classes for advanced
21 typography functionality.

22
23 The data and XML namespace 204 is composed of two namespaces:
24
25

- A data namespace 340 (“System.Data”) containing classes that enable developers to build components that efficiently manage data from multiple data sources. It implements an architecture that, in a disconnected scenario (such as the Internet), provides tools to request, update, and reconcile data in multiple tier systems. The data namespace 340 includes a common namespace 342 that contains types shared by data providers. A data provider describes a collection of types used to access a data source, such as a database, in the managed space. The data namespace 340 also includes an OLE DB namespace 344 that contains types pertaining to data used in object-oriented databases (e.g., Microsoft’s SQL Server), and a SQL client namespace 346 that contains types pertaining to data used by SQL clients. The data namespace also includes a SQL types namespace 348 (“System.Data.SqlTypes”) that contains classes for native data types within Microsoft’s SQL Server. The classes provide a safer, faster alternative to other data types. Using the objects within this namespace helps prevent type conversion errors caused in situations where loss of precision could occur. Because other data types are converted to and from SQL types behind the scenes, explicitly creating and using objects within this namespace results in faster code as well.
- An XML namespace 350 (“System.XML”) containing classes that provide standards-based support for processing XML. The supported standards include XML (e.g., version 1.0), XML Namespaces (both stream level and DOM), XML Schemas, XPath expressions, XSL/T

transformations, DOM Level 2 Core, and SOAP (e.g., version 1.1). The XML namespace 350 includes an XSLT namespace 352 ("System.XML.Xsl") that contains classes and enumerations to support XSLT (Extensible Stylesheet Language Transformations), an Xpath namespace 354 ("System.XML.Xpath") that contains an XPath parser and evaluation engine, and a serialization namespace 356 ("System.XML.Serialization") that contains classes used to serialize objects into XML format documents or streams.

The base class library namespace 206 ("System") includes the following namespaces:

- A collections namespace 360 ("System.Collections") containing interfaces and classes that define various collections of objects, such as lists, queues, arrays, hash tables and dictionaries.
- A configuration namespace 362 ("System.Configuration") containing classes and interfaces that allow developers to programmatically access configuration settings and handle errors in configuration files.
- A diagnostics namespace 364 ("System.Diagnostics") containing classes that are used to debug applications and to trace code execution. The namespace allows developers to start system processes, read and write to event logs, and monitor system performance using performance counters.

- A globalization namespace 366 ("System.Globalization") containing classes that define culture-related information, including the language, the country/region, the calendars in use, the format patterns for dates, currency and numbers, and the sort order for strings.
- An I/O namespace 368 ("System.IO") containing the infrastructure pieces to operate with the input/output of data streams, files, and directories. This namespace includes a model for working with streams of bytes, higher level readers and writers which consume those bytes, various constructions or implementations of the streams (e.g., FileStream and MemoryStream) and, a set of utility classes for working with files and directories.
- A net namespace 370 ("System.Net") providing an extensive set of classes for building network-enabled application, referred to as the Net Class Libraries (NCL). One element to the design of the Net Class Libraries is an extensible, layered approach to exposing networking functionality. The NCL stack contains three basic layers. A base layer (System.Net.Socket) provides access to an interface to TCP/IP, the communications protocol of UNIX networks and the Internet. One example of such an interface is the "WinSock API" from Microsoft Corporation. The next layer is the Transport Protocol classes, which support such transport protocols as TCP and UDP. Developers may write their own protocol classes to provide support for protocols such as IGMP and ICMP. The third layer is the Web request, which provides an abstract factory pattern for the

creation of other protocol classes. The NCL provides implementations for Hyper Text Transport Protocol (HTTP).

- A reflection namespace (“System.Reflection”) 372 containing types that provide a managed view of loaded types, methods, and fields, with the ability to dynamically create and invoke types.
- A resources namespace 374 (“System.Resources”) containing classes and interfaces that allow developers to create, store and manage various culture-specific resources used in an application.
- A security namespace 376 (“System.Security”) supporting the underlying structure of the security system, including interfaces, attributes, exceptions, and base classes for permissions.
- A service process namespace 378 (“System.ServiceProcess”) containing classes that allow developers to install and run services. Services are long-running executables that run without a user interface. They can be installed to run under a system account that enables them to be started at computer reboot. Services whose implementation is derived from processing in one class can define specific behavior for start, stop, pause, and continue commands, as well as behavior to take when the system shuts down.
- A text namespace 380 (“System.Text”) containing classes representing various types of encodings (e.g., ASCII, Unicode, UTF-7, and UTF-8), abstract base classes for converting blocks of characters to and from blocks of bytes, and a helper class that manipulates and formats string objects without creating intermediate instances.

- A threading namespace 382 (“System.Threading”) containing classes and interfaces that enable multi-threaded programming. The threading namespace includes a ThreadPool class that manages groups of threads, a Timer class that enables a delegate to be called after a specified amount of time, and a Mutex class for synchronizing mutually-exclusive threads. This namespace also provides classes for thread scheduling, wait notification, and deadlock resolution.
- A runtime namespace 384 (“System.Runtime”) containing multiple namespaces concerning runtime features, including an interoperation services namespace 386 (“System.Runtime.InteropServices”) that contains a collection of classes useful for accessing COM objects. The types in the InteropServices namespace fall into the following areas of functionality: attributes, exceptions, managed definitions of COM types, wrappers, type converters, and the Marshal class. The runtime namespace 384 further includes a remoting namespace 388 (“System.Runtime.Remoting”) that contains classes and interfaces allowing developers to create and configure distributed applications. Another namespace within the runtime namespace 384 is a serialization namespace 390 (“System.Runtime.Serialization”) that contains classes used for serializing and deserializing objects. Serialization is the process of converting an object or a graph of objects into a linear sequence of bytes for either storage or transmission to another location.

1 Portions of the base class library namespace 206 (“System”) are discussed
2 in additional detail below.

3 4 SYSTEM NAMESPACE

5 The System namespace is the root namespace; it offers common
6 functionality that is needed by a wide variety of application types. The System
7 namespace includes common base classes, types and utility classes that will be
8 needed in substantially all applications (that is, in nearly every application).

9 The System namespace provides commonly used base types. It includes
10 Object, which is the ultimate base class for all types in the system. Object defines
11 the base set of services that any type in the system is able to provide. Not
12 surprisingly, Object provides default implementations for all of these services.
13 The ValueType class is a reference type that serves as the base class for all value
14 types. It customizes the implementations of the virtual methods on Object so that
15 they are more appropriate for value types. Enum is a reference type that derives
16 from ValueType and is the base class for all enums in the system. It further
17 customizes the virtual methods from ValueType to make them specific to deal
18 with exactly one integral field of instance data. Enum also offers utility methods
19 for formatting and parsing of enum values. The ultimate base class for all
20 exceptions in the system, the Exception class, is also in the System namespace.
21 All custom attributes derive from the Attribute base class that contains utility
22 methods for reading custom attribute off of reflection elements.

23 In addition, the base data types are also found in the system namespace.
24 These are types that are so commonly used that languages typically use key words
25

as aliases for them. These classes represent those types and provide formatting and parsing, comparing and coercion support.

NET Framework built-in value types

Category	Class name	Description	Visual Basic data type	C# data type	Managed Extensions for C++ data type
Integer	Byte	An 8-bit unsigned integer.	Byte	byte	char
	Sbyte	An 8-bit signed integer. Not CLS compliant.	Sbyte No built-in type.	sbyte	signed char
	Int16	A 16-bit signed integer.	Short	short	short
	Int32	A 32-bit signed integer.	Integer	int	int -or- long
	Int64	A 64-bit signed integer.	Long	long	__int64
	UInt16	A 16-bit unsigned integer. Not CLS compliant.	UInt16 No built-in type.	ushort	unsigned short
	UInt32	A 32-bit unsigned integer. Not CLS compliant.	UInt32 No built-in type.	uint	unsigned int -or- unsigned long
	UInt64	A 64-bit unsigned integer. Not CLS compliant.	UInt64 No built-in type.	ulong	unsigned __int64
	Single	A single-precision (32-bit) floating-point number.	Single	float	float
	Double	A double-precision (64-bit) floating-point number.	Double	double	double

	Double	A double-precision (64-bit) floating-point number.	Double	double	double
Logical	Boolean	A Boolean value (true or false).	Boolean	bool	bool
Other	Char	A Unicode (16-bit) character.	Char	char	__wchar_t
	Decimal	A 96-bit decimal value.	Decimal	decimal	Decimal
	Single	A signed integer, that is, a 32 bit value on a 32-bit platform and a 64 bit value on a 64-bit platform.	IntPtr No built-in type.	IntPtr No built-in type.	IntPtr No built-in type.
	Double	A native-sized unsigned integer. Not CLS compliant.	UIntPtr No built-in type.	UIntPtr No built-in type.	UIntPtr No built-in type.

Other classes provide services including supervision of managed and unmanaged applications, mathematics, remote and local program invocation, data type conversion, and application environment management.

The following is a more detailed description of the System namespace, identifying various classes, interfaces, enumerations, and so forth contained in the System namespace.

System

This namespace contains fundamental classes and base classes that define commonly-used value and reference data types, events and event handlers, interfaces, attributes, and processing exceptions.

1
2 *Description*

3 This namespace contains fundamental classes and base classes that define
4 commonly-used value and reference data types, events and event handlers,
5 interfaces, attributes, and processing exceptions.

6 _AppDomain interface (System)

7
8
9 *Description*

10
11 Properties:

12 BaseDirectory

13
14 [C#] string BaseDirectory {get;}

15 [C++] String* get_BaseDirectory();

16 [VB] ReadOnly Property BaseDirectory As String

17 [JScript] abstract function get BaseDirectory() : String;

18
19 *Description*

20
21 DynamicDirectory

22
23 [C#] string DynamicDirectory {get;}

24 [C++] String* get_DynamicDirectory();

25 [VB] ReadOnly Property DynamicDirectory As String

1 [JScript] abstract function get DynamicDirectory() : String;

2
3 *Description*

4
5 Evidence

6
7 [C#] Evidence Evidence {get;}

8 [C++] Evidence* get_Evidence();

9 [VB] ReadOnly Property Evidence As Evidence

10 [JScript] abstract function get Evidence() : Evidence;

11
12 *Description*

13
14 FriendlyName

15
16 [C#] string FriendlyName {get;}

17 [C++] String* get_FriendlyName();

18 [VB] ReadOnly Property FriendlyName As String

19 [JScript] abstract function get FriendlyName() : String;

20
21 *Description*

22
23 RelativeSearchPath

24
25 [C#] string RelativeSearchPath {get;}

1 [C++] String* get_RelativeSearchPath();

2 [VB] ReadOnly Property RelativeSearchPath As String

3 [JScript] abstract function get RelativeSearchPath() : String;

5 *Description*

7 ShadowCopyFiles

9 [C#] bool ShadowCopyFiles {get;}

10 [C++] bool get_ShadowCopyFiles();

11 [VB] ReadOnly Property ShadowCopyFiles As Boolean

12 [JScript] abstract function get ShadowCopyFiles() : Boolean;

14 *Description*

17 [C#] event AssemblyLoadEventHandler AssemblyLoad;

18 [C++] __event AssemblyLoadEventHandler* AssemblyLoad;

19 [VB] Event AssemblyLoad As AssemblyLoadEventHandler

21 *Description*

24 [C#] event ResolveEventHandler AssemblyResolve;

25 [C++] __event ResolveEventHandler* AssemblyResolve;

1 [VB] Event AssemblyResolve As ResolveEventHandler

2
3 *Description*
4
5

6 [C#] event EventHandler DomainUnload;

7 [C++] __event EventHandler* DomainUnload;

8 [VB] Event DomainUnload As EventHandler

9
10 *Description*
11
12

13 [C#] event EventHandler ProcessExit;

14 [C++] __event EventHandler* ProcessExit;

15 [VB] Event ProcessExit As EventHandler

16
17 *Description*
18
19

20 [C#] event ResolveEventHandler ResourceResolve;

21 [C++] __event ResolveEventHandler* ResourceResolve;

22 [VB] Event ResourceResolve As ResolveEventHandler

23
24 *Description*
25

1
2 [C#] event ResolveEventHandler TypeResolve;
3 [C++] __event ResolveEventHandler* TypeResolve;
4 [VB] Event TypeResolve As ResolveEventHandler
5

6 *Description*
7
8

9 [C#] event UnhandledExceptionHandler UnhandledException;
10 [C++] __event UnhandledExceptionHandler* UnhandledException;
11 [VB] Event UnhandledException As UnhandledExceptionHandler
12

13 *Description*
14

15 **Methods:**

16 **AppendPrivatePath**
17

18 [C#] void AppendPrivatePath(string path);
19 [C++] void AppendPrivatePath(String* path);
20 [VB] Sub AppendPrivatePath(ByVal path As String)
21 [JScript] function AppendPrivatePath(path : String);
22

23 *Description*
24

25 **ClearPrivatePath**

```

1
2 [C#] void ClearPrivatePath();
3 [C++] void ClearPrivatePath();
4 [VB] Sub ClearPrivatePath()
5 [JScript] function ClearPrivatePath();

```

Description

ClearShadowCopyPath

```

11 [C#] void ClearShadowCopyPath();
12 [C++] void ClearShadowCopyPath();
13 [VB] Sub ClearShadowCopyPath()
14 [JScript] function ClearShadowCopyPath();

```

Description

CreateInstance

```

20 [C#] ObjectHandle CreateInstance(string assemblyName, string typeName);
21 [C++] ObjectHandle* CreateInstance(String* assemblyName, String* typeName);
22 [VB] Function CreateInstance(ByVal assemblyName As String, ByVal typeName
23 As String) As ObjectHandle
24 [JScript] function CreateInstance(assemblyName : String, typeName : String) :
25 ObjectHandle;

```

1
2 *Description*

3
4 **CreateInstance**

5
6 [C#] ObjectHandle CreateInstance(string assemblyName, string typeName,
7 object[] activationAttributes);
8 [C++] ObjectHandle* CreateInstance(String* assemblyName, String* typeName,
9 Object* activationAttributes __gc[]);
10 [VB] Function CreateInstance(ByVal assemblyName As String, ByVal typeName
11 As String, ByVal activationAttributes() As Object) As ObjectHandle
12 [JScript] function CreateInstance(assemblyName : String, typeName : String,
13 activationAttributes : Object[]) : ObjectHandle;

14
15 *Description*

16
17 **CreateInstance**

18
19 [C#] ObjectHandle CreateInstance(string assemblyName, string typeName, bool
20 ignoreCase, BindingFlags bindingAttr, Binder binder, object[] args, CultureInfo
21 culture, object[] activationAttributes, Evidence securityAttributes);
22 [C++] ObjectHandle* CreateInstance(String* assemblyName, String* typeName,
23 bool ignoreCase, BindingFlags bindingAttr, Binder* binder, Object* args __gc[],
24 CultureInfo* culture, Object* activationAttributes __gc[], Evidence*
25 securityAttributes);

1 [VB] Function CreateInstance(ByVal assemblyName As String, ByVal typeName
 2 As String, ByVal ignoreCase As Boolean, ByVal bindingAttr As BindingFlags,
 3 ByVal binder As Binder, ByVal args() As Object, ByVal culture As CultureInfo,
 4 ByVal activationAttributes() As Object, ByVal securityAttributes As Evidence)
 5 As ObjectHandle

6 [JScript] function CreateInstance(assemblyName : String, typeName : String,
 7 ignoreCase : Boolean, bindingAttr : BindingFlags, binder : Binder, args : Object[],
 8 culture : CultureInfo, activationAttributes : Object[], securityAttributes :
 9 Evidence) : ObjectHandle;

Description

CreateInstanceFrom

15 [C#] ObjectHandle CreateInstanceFrom(string assemblyFile, string typeName);

16 [C++] ObjectHandle* CreateInstanceFrom(String* assemblyFile, String*
 17 typeName);

18 [VB] Function CreateInstanceFrom(ByVal assemblyFile As String, ByVal
 19 typeName As String) As ObjectHandle

20 [JScript] function CreateInstanceFrom(assemblyFile : String, typeName : String) :
 21 ObjectHandle;

Description

CreateInstanceFrom


```

1
2 [C#] ObjectHandle CreateInstanceFrom(string assemblyFile, string typeName,
3 object[] activationAttributes);
4 [C++] ObjectHandle* CreateInstanceFrom(String* assemblyFile, String*
5 typeName, Object* activationAttributes __gc[]);
6 [VB] Function CreateInstanceFrom(ByVal assemblyFile As String, ByVal
7 typeName As String, ByVal activationAttributes() As Object) As ObjectHandle
8 [JScript] function CreateInstanceFrom(assemblyFile : String, typeName : String,
9 activationAttributes : Object[]) : ObjectHandle;
10

```

Description

CreateInstanceFrom

```

15 [C#] ObjectHandle CreateInstanceFrom(string assemblyFile, string typeName,
16 bool ignoreCase, BindingFlags bindingAttr, Binder binder, object[] args,
17 CultureInfo culture, object[] activationAttributes, Evidence securityAttributes);
18 [C++] ObjectHandle* CreateInstanceFrom(String* assemblyFile, String*
19 typeName, bool ignoreCase, BindingFlags bindingAttr, Binder* binder, Object*
20 args __gc[], CultureInfo* culture, Object* activationAttributes __gc[], Evidence*
21 securityAttributes);
22 [VB] Function CreateInstanceFrom(ByVal assemblyFile As String, ByVal
23 typeName As String, ByVal ignoreCase As Boolean, ByVal bindingAttr As
24 BindingFlags, ByVal binder As Binder, ByVal args() As Object, ByVal culture As
25 CultureInfo, ByVal activationAttributes() As Object, ByVal securityAttributes As

```

Evidence) As ObjectHandle

[JScript] function CreateInstanceFrom(assemblyFile : String, typeName : String,
ignoreCase : Boolean, bindingAttr : BindingFlags, binder : Binder, args : Object[],
culture : CultureInfo, activationAttributes : Object[], securityAttributes :
Evidence) : ObjectHandle;

Description

DefineDynamicAssembly

[C#] AssemblyBuilder DefineDynamicAssembly(AssemblyName name,
AssemblyBuilderAccess access);
[C++] AssemblyBuilder* DefineDynamicAssembly(AssemblyName* name,
AssemblyBuilderAccess access);
[VB] Function DefineDynamicAssembly(ByVal name As AssemblyName, ByVal
access As AssemblyBuilderAccess) As AssemblyBuilder
[JScript] function DefineDynamicAssembly(name : AssemblyName, access :
AssemblyBuilderAccess) : AssemblyBuilder;

Description

DefineDynamicAssembly

[C#] AssemblyBuilder DefineDynamicAssembly(AssemblyName name,
AssemblyBuilderAccess access, Evidence evidence);

```

1 [C++] AssemblyBuilder* DefineDynamicAssembly(AssemblyName* name,
2 AssemblyBuilderAccess access, Evidence* evidence);
3 [VB] Function DefineDynamicAssembly(ByVal name As AssemblyName, ByVal
4 access As AssemblyBuilderAccess, ByVal evidence As Evidence) As
5 AssemblyBuilder
6 [JScript] function DefineDynamicAssembly(name : AssemblyName, access :
7 AssemblyBuilderAccess, evidence : Evidence) : AssemblyBuilder;
8

```

Description

DefineDynamicAssembly

```

13 [C#] AssemblyBuilder DefineDynamicAssembly(AssemblyName name,
14 AssemblyBuilderAccess access, string dir);
15 [C++] AssemblyBuilder* DefineDynamicAssembly(AssemblyName* name,
16 AssemblyBuilderAccess access, String* dir);
17 [VB] Function DefineDynamicAssembly(ByVal name As AssemblyName, ByVal
18 access As AssemblyBuilderAccess, ByVal dir As String) As AssemblyBuilder
19 [JScript] function DefineDynamicAssembly(name : AssemblyName, access :
20 AssemblyBuilderAccess, dir : String) : AssemblyBuilder;
21

```

Description

DefineDynamicAssembly

```

1
2 [C#] AssemblyBuilder DefineDynamicAssembly(AssemblyName name,
3 AssemblyBuilderAccess access, string dir, Evidence evidence);
4 [C++] AssemblyBuilder* DefineDynamicAssembly(AssemblyName* name,
5 AssemblyBuilderAccess access, String* dir, Evidence* evidence);
6 [VB] Function DefineDynamicAssembly(ByVal name As AssemblyName, ByVal
7 access As AssemblyBuilderAccess, ByVal dir As String, ByVal evidence As
8 Evidence) As AssemblyBuilder
9 [JScript] function DefineDynamicAssembly(name : AssemblyName, access :
10 AssemblyBuilderAccess, dir : String, evidence : Evidence) : AssemblyBuilder;
11

```

Description

DefineDynamicAssembly

```

16 [C#] AssemblyBuilder DefineDynamicAssembly(AssemblyName name,
17 AssemblyBuilderAccess access, PermissionSet requiredPermissions,
18 PermissionSet optionalPermissions, PermissionSet refusedPermissions);
19 [C++] AssemblyBuilder* DefineDynamicAssembly(AssemblyName* name,
20 AssemblyBuilderAccess access, PermissionSet* requiredPermissions,
21 PermissionSet* optionalPermissions, PermissionSet* refusedPermissions);
22 [VB] Function DefineDynamicAssembly(ByVal name As AssemblyName, ByVal
23 access As AssemblyBuilderAccess, ByVal requiredPermissions As PermissionSet,
24 ByVal optionalPermissions As PermissionSet, ByVal refusedPermissions As
25 PermissionSet) As AssemblyBuilder

```

1 [JScript] function DefineDynamicAssembly(name : AssemblyName, access :
2 AssemblyBuilderAccess, requiredPermissions : PermissionSet,
3 optionalPermissions : PermissionSet, refusedPermissions : PermissionSet) :
4 AssemblyBuilder;

6 *Description*

8 DefineDynamicAssembly

10 [C#] AssemblyBuilder DefineDynamicAssembly(AssemblyName name,
11 AssemblyBuilderAccess access, Evidence evidence, PermissionSet
12 requiredPermissions, PermissionSet optionalPermissions, PermissionSet
13 refusedPermissions);

14 [C++] AssemblyBuilder* DefineDynamicAssembly(AssemblyName* name,
15 AssemblyBuilderAccess access, Evidence* evidence, PermissionSet*
16 requiredPermissions, PermissionSet* optionalPermissions, PermissionSet*
17 refusedPermissions);

18 [VB] Function DefineDynamicAssembly(ByVal name As AssemblyName, ByVal
19 access As AssemblyBuilderAccess, ByVal evidence As Evidence, ByVal
20 requiredPermissions As PermissionSet, ByVal optionalPermissions As
21 PermissionSet, ByVal refusedPermissions As PermissionSet) As AssemblyBuilder

22 [JScript] function DefineDynamicAssembly(name : AssemblyName, access :
23 AssemblyBuilderAccess, evidence : Evidence, requiredPermissions :
24 PermissionSet, optionalPermissions : PermissionSet, refusedPermissions :
25 PermissionSet) : AssemblyBuilder;

Description

DefineDynamicAssembly

[C#] AssemblyBuilder DefineDynamicAssembly(AssemblyName name, AssemblyBuilderAccess access, string dir, PermissionSet requiredPermissions, PermissionSet optionalPermissions, PermissionSet refusedPermissions);

[C++] AssemblyBuilder* DefineDynamicAssembly(AssemblyName* name, AssemblyBuilderAccess access, String* dir, PermissionSet* requiredPermissions, PermissionSet* optionalPermissions, PermissionSet* refusedPermissions);

[VB] Function DefineDynamicAssembly(ByVal name As AssemblyName, ByVal access As AssemblyBuilderAccess, ByVal dir As String, ByVal requiredPermissions As PermissionSet, ByVal optionalPermissions As PermissionSet, ByVal refusedPermissions As PermissionSet) As AssemblyBuilder

[JScript] function DefineDynamicAssembly(name : AssemblyName, access : AssemblyBuilderAccess, dir : String, requiredPermissions : PermissionSet, optionalPermissions : PermissionSet, refusedPermissions : PermissionSet) : AssemblyBuilder;

Description

DefineDynamicAssembly

[C#] AssemblyBuilder DefineDynamicAssembly(AssemblyName name,

```

1  AssemblyBuilderAccess access, string dir, Evidence evidence, PermissionSet
2  requiredPermissions, PermissionSet optionalPermissions, PermissionSet
3  refusedPermissions);
4  [C++] AssemblyBuilder* DefineDynamicAssembly(AssemblyName* name,
5  AssemblyBuilderAccess access, String* dir, Evidence* evidence, PermissionSet*
6  requiredPermissions, PermissionSet* optionalPermissions, PermissionSet*
7  refusedPermissions);
8  [VB] Function DefineDynamicAssembly(ByVal name As AssemblyName, ByVal
9  access As AssemblyBuilderAccess, ByVal dir As String, ByVal evidence As
10 Evidence, ByVal requiredPermissions As PermissionSet, ByVal
11 optionalPermissions As PermissionSet, ByVal refusedPermissions As
12 PermissionSet) As AssemblyBuilder
13 [JScript] function DefineDynamicAssembly(name : AssemblyName, access :
14 AssemblyBuilderAccess, dir : String, evidence : Evidence, requiredPermissions :
15 PermissionSet, optionalPermissions : PermissionSet, refusedPermissions :
16 PermissionSet) : AssemblyBuilder;

```

Description

DefineDynamicAssembly

```

22 [C#] AssemblyBuilder DefineDynamicAssembly(AssemblyName name,
23 AssemblyBuilderAccess access, string dir, Evidence evidence, PermissionSet
24 requiredPermissions, PermissionSet optionalPermissions, PermissionSet
25 refusedPermissions, bool isSynchronized);

```

1 [C++] AssemblyBuilder* DefineDynamicAssembly(AssemblyName* name,
2 AssemblyBuilderAccess access, String* dir, Evidence* evidence, PermissionSet*
3 requiredPermissions, PermissionSet* optionalPermissions, PermissionSet*
4 refusedPermissions, bool isSynchronized);

5 [VB] Function DefineDynamicAssembly(ByVal name As AssemblyName, ByVal
6 access As AssemblyBuilderAccess, ByVal dir As String, ByVal evidence As
7 Evidence, ByVal requiredPermissions As PermissionSet, ByVal
8 optionalPermissions As PermissionSet, ByVal refusedPermissions As
9 PermissionSet, ByVal isSynchronized As Boolean) As AssemblyBuilder

10 [JScript] function DefineDynamicAssembly(name : AssemblyName, access :
11 AssemblyBuilderAccess, dir : String, evidence : Evidence, requiredPermissions :
12 PermissionSet, optionalPermissions : PermissionSet, refusedPermissions :
13 PermissionSet, isSynchronized : Boolean) : AssemblyBuilder;

15 *Description*

17 DoCallback

19 [C#] void DoCallback(CrossAppDomainDelegate theDelegate);
20 [C++] void DoCallback(CrossAppDomainDelegate* theDelegate);
21 [VB] Sub DoCallback(ByVal theDelegate As CrossAppDomainDelegate)
22 [JScript] function DoCallback(theDelegate : CrossAppDomainDelegate);

24 *Description*

1 Equals

2

3 [C#] bool Equals(object other);

4 [C++] bool Equals(Object* other);

5 [VB] Function Equals(ByVal other As Object) As Boolean

6 [JScript] function Equals(other : Object) : Boolean;

7

8 *Description*

9

10 ExecuteAssembly

11

12 [C#] int ExecuteAssembly(string assemblyFile);

13 [C++] int ExecuteAssembly(String* assemblyFile);

14 [VB] Function ExecuteAssembly(ByVal assemblyFile As String) As Integer

15 [JScript] function ExecuteAssembly(assemblyFile : String) : int;

16

17 *Description*

18

19 ExecuteAssembly

20

21 [C#] int ExecuteAssembly(string assemblyFile, Evidence assemblySecurity);

22 [C++] int ExecuteAssembly(String* assemblyFile, Evidence* assemblySecurity);

23 [VB] Function ExecuteAssembly(ByVal assemblyFile As String, ByVal

24 assemblySecurity As Evidence) As Integer

25 [JScript] function ExecuteAssembly(assemblyFile : String, assemblySecurity :

Evidence) : int;

Description

ExecuteAssembly

[C#] int ExecuteAssembly(string assemblyFile, Evidence assemblySecurity,
string[] args);

[C++] int ExecuteAssembly(String* assemblyFile, Evidence* assemblySecurity,
String* args __gc[]);

[VB] Function ExecuteAssembly(ByVal assemblyFile As String, ByVal
assemblySecurity As Evidence, ByVal args() As String) As Integer

[JScript] function ExecuteAssembly(assemblyFile : String, assemblySecurity :
Evidence, args : String[]) : int;

Description

GetAssemblies

[C#] Assembly[] GetAssemblies();

[C++] Assembly* GetAssemblies() [];

[VB] Function GetAssemblies() As Assembly()

[JScript] function GetAssemblies() : Assembly[];

Description

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

GetData

[C#] object GetData(string name);
[C++] Object* GetData(String* name);
[VB] Function GetData(ByVal name As String) As Object
[JScript] function GetData(name : String) : Object;

Description

GetHashCode

[C#] int GetHashCode();
[C++] int GetHashCode();
[VB] Function GetHashCode() As Integer
[JScript] function GetHashCode() : int;

Description

GetLifetimeService

[C#] object GetLifetimeService();
[C++] Object* GetLifetimeService();
[VB] Function GetLifetimeService() As Object
[JScript] function GetLifetimeService() : Object;

Description

GetType

[C#] Type GetType();

[C++] Type* GetType();

[VB] Function GetType() As Type

[JScript] function GetType() : Type;

Description

InitializeLifetimeService

[C#] object InitializeLifetimeService();

[C++] Object* InitializeLifetimeService();

[VB] Function InitializeLifetimeService() As Object

[JScript] function InitializeLifetimeService() : Object;

Description

Load

[C#] Assembly Load(AssemblyName assemblyRef);

[C++] Assembly* Load(AssemblyName* assemblyRef);

1 [VB] Function Load(ByVal assemblyRef As AssemblyName) As Assembly

2 [JScript] function Load(assemblyRef : AssemblyName) : Assembly;

3

4 *Description*

5

6 Load

7

8 [C#] Assembly Load(byte[] rawAssembly);

9 [C++] Assembly* Load(unsigned char rawAssembly __gc[]);

10 [VB] Function Load(ByVal rawAssembly() As Byte) As Assembly

11 [JScript] function Load(rawAssembly : Byte[]) : Assembly;

12

13 *Description*

14

15 Load

16

17 [C#] Assembly Load(string assemblyString);

18 [C++] Assembly* Load(String* assemblyString);

19 [VB] Function Load(ByVal assemblyString As String) As Assembly

20 [JScript] function Load(assemblyString : String) : Assembly;

21

22 *Description*

23

24 Load

25

```

1
2 [C#] Assembly Load(AssemblyName assemblyRef, Evidence assemblySecurity);
3 [C++] Assembly* Load(AssemblyName* assemblyRef, Evidence*
4 assemblySecurity);
5 [VB] Function Load(ByVal assemblyRef As AssemblyName, ByVal
6 assemblySecurity As Evidence) As Assembly
7 [JScript] function Load(assemblyRef : AssemblyName, assemblySecurity :
8 Evidence) : Assembly;
9

```

Description

Load

```

13
14 [C#] Assembly Load(byte[] rawAssembly, byte[] rawSymbolStore);
15 [C++] Assembly* Load(unsigned char rawAssembly __gc[], unsigned char
16 rawSymbolStore __gc[]);
17 [VB] Function Load(ByVal rawAssembly() As Byte, ByVal rawSymbolStore() As
18 Byte) As Assembly
19 [JScript] function Load(rawAssembly : Byte[], rawSymbolStore : Byte[]) :
20 Assembly;
21

```

Description

Load

```

1
2 [C#] Assembly Load(string assemblyString, Evidence assemblySecurity);
3 [C++] Assembly* Load(String* assemblyString, Evidence* assemblySecurity);
4 [VB] Function Load(ByVal assemblyString As String, ByVal assemblySecurity
5 As Evidence) As Assembly
6 [JScript] function Load(assemblyString : String, assemblySecurity : Evidence) :
7 Assembly;
8

```

Description

Load

```

13 [C#] Assembly Load(AssemblyName assemblyRef, Evidence assemblySecurity,
14 string callerLocation);
15 [C++] Assembly* Load(AssemblyName* assemblyRef, Evidence*
16 assemblySecurity, String* callerLocation);
17 [VB] Function Load(ByVal assemblyRef As AssemblyName, ByVal
18 assemblySecurity As Evidence, ByVal callerLocation As String) As Assembly
19 [JScript] function Load(assemblyRef : AssemblyName, assemblySecurity :
20 Evidence, callerLocation : String) : Assembly;
21

```

Description

Load

```

1
2 [C#] Assembly Load(byte[] rawAssembly, byte[] rawSymbolStore, Evidence
3 securityEvidence);
4 [C++] Assembly* Load(unsigned char rawAssembly __gc[], unsigned char
5 rawSymbolStore __gc[], Evidence* securityEvidence);
6 [VB] Function Load(ByteVal rawAssembly() As Byte, ByteVal rawSymbolStore() As
7 Byte, ByteVal securityEvidence As Evidence) As Assembly
8 [JScript] function Load(rawAssembly : Byte[], rawSymbolStore : Byte[],
9 securityEvidence : Evidence) : Assembly;
10

```

Description

Load

```

15 [C#] Assembly Load(string assemblyString, Evidence assemblySecurity, string
16 callerLocation);
17 [C++] Assembly* Load(String* assemblyString, Evidence* assemblySecurity,
18 String* callerLocation);
19 [VB] Function Load(ByteVal assemblyString As String, ByteVal assemblySecurity
20 As Evidence, ByteVal callerLocation As String) As Assembly
21 [JScript] function Load(assemblyString : String, assemblySecurity : Evidence,
22 callerLocation : String) : Assembly;
23

```

Description

SetAppDomainPolicy

```
[C#] void SetAppDomainPolicy(PolicyLevel domainPolicy);  
[C++] void SetAppDomainPolicy(PolicyLevel* domainPolicy);  
[VB] Sub SetAppDomainPolicy(ByVal domainPolicy As PolicyLevel)  
[JScript] function SetAppDomainPolicy(domainPolicy : PolicyLevel);
```

Description

SetCachePath

```
[C#] void SetCachePath(string s);  
[C++] void SetCachePath(String* s);  
[VB] Sub SetCachePath(ByVal s As String)  
[JScript] function SetCachePath(s : String);
```

Description

SetData

```
[C#] void SetData(string name, object data);  
[C++] void SetData(String* name, Object* data);  
[VB] Sub SetData(ByVal name As String, ByVal data As Object)  
[JScript] function SetData(name : String, data : Object);
```

1
2 *Description*

3
4 SetPrincipalPolicy

5
6 [C#] void SetPrincipalPolicy(PrincipalPolicy policy);

7 [C++] void SetPrincipalPolicy(PrincipalPolicy policy);

8 [VB] Sub SetPrincipalPolicy(ByVal policy As PrincipalPolicy)

9 [JScript] function SetPrincipalPolicy(policy : PrincipalPolicy);

10
11 *Description*

12
13 SetShadowCopyPath

14
15 [C#] void SetShadowCopyPath(string s);

16 [C++] void SetShadowCopyPath(String* s);

17 [VB] Sub SetShadowCopyPath(ByVal s As String)

18 [JScript] function SetShadowCopyPath(s : String);

19
20 *Description*

21
22 SetThreadPrincipal

23
24 [C#] void SetThreadPrincipal(IPrincipal principal);

25 [C++] void SetThreadPrincipal(IPrincipal* principal);

1 [VB] Sub SetThreadPrincipal(ByVal principal As IPincipal)

2 [JScript] function SetThreadPrincipal(principal : IPincipal);

3
4 *Description*

5
6 ToString

7
8 [C#] string ToString();

9 [C++] String* ToString();

10 [VB] Function ToString() As String

11 [JScript] function ToString() : String;

12
13 *Description*

14
15 Activator class (System)

16 ToString

17
18
19 *Description*

20 Contains methods to create types of objects locally or remotely, or obtain
21 references to existing remote objects.

22 The

23 **System.Activator.CreateInstance(System.Type, System.Reflection.BindingFlags,**
24 **System.Reflection.Binder, System.Object[], System.Globalization.CultureInfo)**
25 **fo)** method creates an instance of a type defined in an assembly by invoking the

constructor that best matches the specified arguments. If no arguments are specified, the constructor that takes no parameters, that is, the default constructor, is invoked.

CreateComInstanceFrom

```
[C#] public static ObjectHandle CreateComInstanceFrom(string assemblyName,  
string typeName);
```

```
[C++] public: static ObjectHandle* CreateComInstanceFrom(String*  
assemblyName, String* typeName);
```

```
[VB] Public Shared Function CreateComInstanceFrom(ByVal assemblyName As  
String, ByVal typeName As String) As ObjectHandle
```

```
[JScript] public static function CreateComInstanceFrom(assemblyName : String,  
typeName : String) : ObjectHandle;
```

Description

Creates an instance of the COM object whose name is specified, using the named assembly file and the constructor that best matches the specified parameters.

Return Value: A handle, which must be unwrapped to access the newly created instance.

This method allows types to be created remotely without having to load the type locally. The name of a file that contains an assembly where the type named *typeName* is sought. The name of the desired type.

CreateInstance

```

1
2 [C#] public static object CreateInstance(Type type);
3 [C++] public: static Object* CreateInstance(Type* type);
4 [VB] Public Shared Function CreateInstance(ByVal type As Type) As Object
5 [JScript] public static function CreateInstance(type : Type) : Object;
6

```

Description

Creates an instance of the specified type using the constructor that best matches the specified parameter.

Return Value: A reference to the newly created object.

The constructor to be invoked must be accessible. The type of object to create.

CreateInstance

```

14
15 [C#] public static ObjectHandle CreateInstance(string assemblyName, string
16 typeName);
17 [C++] public: static ObjectHandle* CreateInstance(String* assemblyName,
18 String* typeName);
19 [VB] Public Shared Function CreateInstance(ByVal assemblyName As String,
20 ByVal typeName As String) As ObjectHandle
21 [JScript] public static function CreateInstance(assemblyName : String, typeName :
22 String) : ObjectHandle;
23

```

Description

Creates an instance of the type whose name is specified, using the named assembly and the constructor that best matches the specified parameters.

Return Value: A handle, which must be unwrapped to access the newly created instance.

This method allows types to be created remotely without having to load the type locally. The name of the assembly where the type named *typeName* is sought. If *assemblyName* is **null**, the executing assembly is searched. The name of the desired type.

CreateInstance

[C#] public static object CreateInstance(Type type, object[] args);

[C++] public: static Object* CreateInstance(Type* type, Object* args __gc[]);

[VB] Public Shared Function CreateInstance(ByVal type As Type, ByVal args() As Object) As Object

[JScript] public static function CreateInstance(type : Type, args : Object[]) : Object;

Description

Creates an instance of the specified type using the constructor that best matches the specified parameters.

Return Value: A reference to the newly created object.

The constructor to be invoked must be accessible and provide the most specific match with the specified argument list. The type of object to create. An array of arguments that match in number, order, and type the parameters of the

1 constructor to invoke. If *args* is an empty array or **null**, the constructor that takes
2 no parameters (the default constructor) is invoked.

3 CreateInstance

4
5 [C#] public static ObjectHandle CreateInstance(string assemblyName, string
6 typeName, object[] activationAttributes);

7 [C++] public: static ObjectHandle* CreateInstance(String* assemblyName,
8 String* typeName, Object* activationAttributes __gc[]);

9 [VB] Public Shared Function CreateInstance(ByVal assemblyName As String,
10 ByVal typeName As String, ByVal activationAttributes() As Object) As
11 ObjectHandle

12 [JScript] public static function CreateInstance(assemblyName : String, typeName :
13 String, activationAttributes : Object[]) : ObjectHandle;

15 *Description*

16 Creates an instance of the type whose name is specified, using the named
17 assembly and the constructor that best matches the specified parameters.

18 *Return Value:* A handle, which must be unwrapped to access the newly created
19 instance.

20 This method allows types to be created remotely without having to load the
21 type locally. The name of the assembly where the type named *typeName* is sought.
22 If *assemblyName* is **null**, the executing assembly is searched. The name of the
23 desired type. An array of one or more attributes that can participate in activation.

24 CreateInstance

```

1
2 [C#] public static object CreateInstance(Type type, object[] args, object[]
3 activationAttributes);
4 [C++] public: static Object* CreateInstance(Type* type, Object* args __gc[],
5 Object* activationAttributes __gc[]);
6 [VB] Public Shared Function CreateInstance(ByVal type As Type, ByVal args()
7 As Object, ByVal activationAttributes() As Object) As Object
8 [JScript] public static function CreateInstance(type : Type, args : Object[],
9 activationAttributes : Object[]) : Object;
10

```

Description

Creates an instance of the specified type using the constructor that best matches the specified parameters.

Return Value: A reference to the newly created object.

The constructor to be invoked must be accessible and provide the most specific match with the specified argument list. The type of object to create. An array of arguments that match in number, order, and type the parameters of the constructor to invoke. If *args* is an empty array or **null**, the constructor that takes no parameters (the default constructor) is invoked. An array of one or more attributes that can participate in activation.

CreateInstance

```

22
23 [C#] public static object CreateInstance(Type type, BindingFlags bindingAttr,
24 Binder binder, object[] args, CultureInfo culture);
25 [C++] public: static Object* CreateInstance(Type* type, BindingFlags

```



```

1 bindingAttr, Binder* binder, Object* args __gc[], CultureInfo* culture);
2 [VB] Public Shared Function CreateInstance(ByVal type As Type, ByVal
3 bindingAttr As BindingFlags, ByVal binder As Binder, ByVal args() As Object,
4 ByVal culture As CultureInfo) As Object
5 [JScript] public static function CreateInstance(type : Type, bindingAttr :
6 BindingFlags, binder : Binder, args : Object[], culture : CultureInfo) : Object;
7 Creates an instance of the specified type using the constructor that best matches
8 the specified parameters.

```

Description

Creates an instance of the specified type using the constructor that best matches the specified parameters.

Return Value: A reference to the newly created object.

The constructor to be invoked must be accessible and provide the most specific match with the specified argument list under the constraints of the specified binder and binding attributes. The type of object to create. A combination of zero or more bit flags that affect the search for the *type* constructor. If *bindingAttr* is zero, a case-sensitive search for public properties is conducted. An object that uses *bindingAttr* and *args* to seek and identify the *type* constructor. If *binder* is **null**, the default binder is used. An array of arguments that match in number, order, and type the parameters of the constructor to invoke. If *args* is an empty array or **null**, the constructor that takes no parameters (the default constructor) is invoked. Culture-specific information that governs the coercion of *args* to the formal types declared for the *type* constructor. If *culture* is **null**, the **System.Globalization.CultureInfo** for the current thread is used.

CreateInstance

```
[C#] public static object CreateInstance(Type type, BindingFlags bindingAttr,
Binder binder, object[] args, CultureInfo culture, object[] activationAttributes);
[C++] public: static Object* CreateInstance(Type* type, BindingFlags
bindingAttr, Binder* binder, Object* args __gc[], CultureInfo* culture, Object*
activationAttributes __gc[]);
[VB] Public Shared Function CreateInstance(ByVal type As Type, ByVal
bindingAttr As BindingFlags, ByVal binder As Binder, ByVal args() As Object,
ByVal culture As CultureInfo, ByVal activationAttributes() As Object) As Object
[JScript] public static function CreateInstance(type : Type, bindingAttr :
BindingFlags, binder : Binder, args : Object[], culture : CultureInfo,
activationAttributes : Object[]) : Object;
```

Description

Creates an instance of the specified type using the constructor that best matches the specified parameters.

Return Value: A reference to the newly created object.

The constructor to be invoked must be accessible and provide the most specific match with the specified argument list under the constraints of the specified binder and binding attributes. The type of object to create. A combination of zero or more bit flags that affect the search for the *type* constructor. If *bindingAttr* is zero, a case-sensitive search for public properties is conducted. An object that uses *bindingAttr* and *args* to seek and identify the *type* constructor. If *binder* is **null**, the default binder is used. An array of arguments that

match in number, order, and type the parameters of the constructor to invoke. If *args* is an empty array or **null**, the constructor that takes no parameters (the default constructor) is invoked. Culture-specific information that governs the coercion of *args* to the formal types declared for the *type* constructor. If *culture* is **null**, the **System.Globalization.CultureInfo** for the current thread is used. An array of one or more attributes that can participate in activation.

CreateInstance

```
[C#] public static ObjectHandle CreateInstance(string assemblyName, string
typeName, bool ignoreCase, BindingFlags bindingAttr, Binder binder, object[]
args, CultureInfo culture, object[] activationAttributes, Evidence securityInfo);
[C++] public: static ObjectHandle* CreateInstance(String* assemblyName,
String* typeName, bool ignoreCase, BindingFlags bindingAttr, Binder* binder,
Object* args __gc[], CultureInfo* culture, Object* activationAttributes __gc[],
Evidence* securityInfo);
[VB] Public Shared Function CreateInstance(ByVal assemblyName As String,
ByVal typeName As String, ByVal ignoreCase As Boolean, ByVal bindingAttr As
BindingFlags, ByVal binder As Binder, ByVal args() As Object, ByVal culture As
CultureInfo, ByVal activationAttributes() As Object, ByVal securityInfo As
Evidence) As ObjectHandle
[JScript] public static function CreateInstance(assemblyName : String, typeName :
String, ignoreCase : Boolean, bindingAttr : BindingFlags, binder : Binder, args :
Object[], culture : CultureInfo, activationAttributes : Object[], securityInfo :
Evidence) : ObjectHandle;
```

Description

Creates an instance of the type whose name is specified, using the named assembly and the constructor that best matches the specified parameters.

Return Value: A handle, which must be unwrapped to access the newly created instance.

This method allows types to be created remotely without having to load the type locally. The name of the assembly where the type named *typeName* is sought. If *assemblyName* is **null**, the executing assembly is searched. The name of the desired type. A Boolean that specifies whether the search for *typeName* is case-sensitive. If *ignoreCase* is **true**, the search is not case-sensitive. A combination of zero or more bit flags that affect the search for the *typeName* constructor. If *bindingAttr* is zero, a case-sensitive search for public properties is conducted. An object that uses *bindingAttr* and *args* to seek and identify the *typeName* constructor. If *binder* is **null**, the default binder is used. An array of arguments that match in number, order, and type the parameters of the constructor to invoke. If *args* is an empty array or **null**, the constructor that takes no parameters (the default constructor) is invoked. Culture-specific information that governs the coercion of *args* to the formal types declared for the *typeName* constructor. If *culture* is **null**, the **System.Globalization.CultureInfo** for the current thread is used. An array of one or more attributes that can participate in activation. Information used to make security policy decisions and grant code permissions.

CreateInstanceFrom

[C#] public static ObjectHandle CreateInstanceFrom(string assemblyFile, string

```

1  typeName);
2  [C++] public: static ObjectHandle* CreateInstanceFrom(String* assemblyFile,
3  String* typeName);
4  [VB] Public Shared Function CreateInstanceFrom(ByVal assemblyFile As String,
5  ByVal typeName As String) As ObjectHandle
6  [JScript] public static function CreateInstanceFrom(assemblyFile : String,
7  typeName : String) : ObjectHandle; Creates an instance of the type whose name is
8  specified, using the named assembly file and the constructor that best matches the
9  specified parameters.

```

Description

Creates an instance of the type whose name is specified, using the named assembly file and the constructor that best matches the specified parameters.

Return Value: A handle, which must be unwrapped to access the newly created instance.

This method allows types to be created remotely without having to load the type locally. The name of a file that contains an assembly where the type named *typeName* is sought. The name of the desired type.

CreateInstanceFrom

```

21 [C#] public static ObjectHandle CreateInstanceFrom(string assemblyFile, string
22 typeName, object[] activationAttributes);
23 [C++] public: static ObjectHandle* CreateInstanceFrom(String* assemblyFile,
24 String* typeName, Object* activationAttributes __gc[]);
25 [VB] Public Shared Function CreateInstanceFrom(ByVal assemblyFile As String,

```

ByVal typeName As String, ByVal activationAttributes() As Object) As
ObjectHandle

[JScript] public static function CreateInstanceFrom(assemblyFile : String,
typeName : String, activationAttributes : Object[]) : ObjectHandle;

Description

Creates an instance of the type whose name is specified, using the named
assembly file and the constructor that best matches the specified parameters.

Return Value: A handle, which must be unwrapped to access the newly created
instance.

This method allows types to be created remotely without having to load the
type locally. The name of a file that contains an assembly where the type named
typeName is sought. The name of the desired type. An array of one or more
attributes that can participate in activation.

CreateInstanceFrom

[C#] public static ObjectHandle CreateInstanceFrom(string assemblyFile, string
typeName, bool ignoreCase, BindingFlags bindingAttr, Binder binder, object[]
args, CultureInfo culture, object[] activationAttributes, Evidence securityInfo);

[C++] public: static ObjectHandle* CreateInstanceFrom(String* assemblyFile,
String* typeName, bool ignoreCase, BindingFlags bindingAttr, Binder* binder,
Object* args __gc[], CultureInfo* culture, Object* activationAttributes __gc[],
Evidence* securityInfo);

[VB] Public Shared Function CreateInstanceFrom(ByVal assemblyFile As String,
ByVal typeName As String, ByVal ignoreCase As Boolean, ByVal bindingAttr As

```

1 BindingFlags, ByVal binder As Binder, ByVal args() As Object, ByVal culture As
2 CultureInfo, ByVal activationAttributes() As Object, ByVal securityInfo As
3 Evidence) As ObjectHandle
4 [JScript] public static function CreateInstanceFrom(assemblyFile : String,
5 typeName : String, ignoreCase : Boolean, bindingAttr : BindingFlags, binder :
6 Binder, args : Object[], culture : CultureInfo, activationAttributes : Object[],
7 securityInfo : Evidence) : ObjectHandle;
8

```

Description

Creates an instance of the type whose name is specified, using the named assembly file and the constructor that best matches the specified parameters.

Return Value: A handle, which must be unwrapped to access the newly created instance.

This method allows types to be created remotely without having to load the type locally. The name of a file that contains an assembly where the type named *typeName* is sought. The name of the desired type. A Boolean that specifies whether the search for *typeName* is case-sensitive. If *ignoreCase* is **true**, the search is not case-sensitive. A combination of zero or more bit flags that affect the search for the *typeName* constructor. If *bindingAttr* is zero, a case-sensitive search for public properties is conducted. An object that uses *bindingAttr* and *args* to seek and identify the *typeName* constructor. If *binder* is **null**, the default binder is used. An array of arguments that match in number, order, and type the parameters of the constructor to invoke. If *args* is an empty array or **null**, the constructor that takes no parameters (the default constructor) is invoked. Culture-specific information that governs the coercion of *args* to the formal types declared for the *typeName*

1 constructor. If *culture* is **null**, the **System.Globalization.CultureInfo** for the
2 current thread is used. An array of one or more attributes that can participate in
3 activation. Information used to make security policy decisions and grant code
4 permissions.

5 GetObject

6
7 [C#] public static object GetObject(Type type, string url);

8 [C++] public: static Object* GetObject(Type* type, String* url);

9 [VB] Public Shared Function GetObject(ByVal type As Type, ByVal url As
10 String) As Object

11 [JScript] public static function GetObject(type : Type, url : String) : Object;

12 Creates a proxy for a currently running remote object, server-activated well-
13 known object, or web service.

14 15 *Description*

16 Creates a proxy for the well-known object indicated by the specified type
17 and URL.

18 *Return Value:* A proxy that points to an endpoint served by the requested well-
19 known object.

20 Call the proxy to send messages to the remote object. No messages are sent
21 over the network until a method is called on the proxy. The type of the well-
22 known object to which you want to connect. The URL of the well-known object.

23 GetObject

24
25 [C#] public static object GetObject(Type type, string url, object state);


```

1 [C++] public: static Object* GetObject(Type* type, String* url, Object* state);
2 [VB] Public Shared Function GetObject(ByVal type As Type, ByVal url As
3 String, ByVal state As Object) As Object
4 [JScript] public static function GetObject(type : Type, url : String, state : Object) :
5 Object;
6

```

Description

Creates a proxy for the well-known object indicated by the specified type, URL, and channel data.

Return Value: A proxy that points to an endpoint served by the requested well-known object.

Call the proxy to send messages to the remote object. No messages are sent over the network until a method is called on the proxy. The type of the well-known object to which you want to connect. The URL of the well-known object. Channel-specific data or **null**.

AppDomain class (System)

ToString

Description

Represents an application domain, which is an isolated environment where applications execute. This class cannot be inherited.

Application domains isolate executing applications from one another. One or more applications can run in a single application domain.

BaseDirectory

ToString

[C#] public string BaseDirectory {get;}

[C++] public: __property String* get_BaseDirectory();

[VB] Public ReadOnly Property BaseDirectory As String

[JScript] public function get BaseDirectory() : String;

Description

Gets the base directory that the assembly resolver used to probe for assemblies.

This property corresponds to the assembly resolver's APPBASE.

CurrentDomain

ToString

[C#] public static AppDomain CurrentDomain {get;}

[C++] public: __property static AppDomain* get_CurrentDomain();

[VB] Public Shared ReadOnly Property CurrentDomain As AppDomain

[JScript] public static function get CurrentDomain() : AppDomain;

Description

Gets the current application domain for the current

System.Threading.Thread .

DynamicDirectory

ToString

```

1
2 [C#] public string DynamicDirectory {get;}
3 [C++] public: __property String* get_DynamicDirectory();
4 [VB] Public ReadOnly Property DynamicDirectory As String
5 [JScript] public function get DynamicDirectory() : String;
6

```

7 *Description*

8 Gets the directory that the assembly resolver used to probe for dynamically-
9 created assemblies.

10 Only available once an attempt has been made to load an assembly into this
11 domain.

12 Evidence

13 ToString

```

14
15 [C#] public Evidence Evidence {get;}
16 [C++] public: __property Evidence* get_Evidence();
17 [VB] Public ReadOnly Property Evidence As Evidence
18 [JScript] public function get Evidence() : Evidence;
19

```

20 *Description*

21 Gets the **System.Security.Policy.Evidence** associated with this application
22 domain that is used as input to security policy.

23 FriendlyName

24 ToString

```

1
2 [C#] public string FriendlyName {get;}
3 [C++] public: __property String* get_FriendlyName();
4 [VB] Public ReadOnly Property FriendlyName As String
5 [JScript] public function get FriendlyName() : String;
6

```

7 *Description*

8 Gets the friendly name of this application domain.

9 The friendly name of the default application domain is the name of the
10 assembly file loaded in the application domain. The friendly name is formed by
11 stripping the directory specification from the assembly's codebase. For example, if
12 an assembly with the file name "c:\MyAppDirectory\MyAssembly.exe" is loaded
13 in the default application domain, the friendly name of that application domain is
14 "MyAssembly.exe" .

15 RelativeSearchPath

16 ToString

```

17
18 [C#] public string RelativeSearchPath {get;}
19 [C++] public: __property String* get_RelativeSearchPath();
20 [VB] Public ReadOnly Property RelativeSearchPath As String
21 [JScript] public function get RelativeSearchPath() : String;
22

```

23 *Description*

24 Gets the path relative to the base directory where the assembly resolver
25 should probe for private assemblies.

Private assemblies are deployed in the same directory structure as the application.

SetupInformation

ToString

[C#] public AppDomainSetup SetupInformation {get;}

[C++] public: __property AppDomainSetup* get_SetupInformation();

[VB] Public ReadOnly Property SetupInformation As AppDomainSetup

[JScript] public function get SetupInformation() : AppDomainSetup;

Description

Gets the application domain configuration information for this instance.

ShadowCopyFiles

ToString

[C#] public bool ShadowCopyFiles {get;}

[C++] public: __property bool get_ShadowCopyFiles();

[VB] Public ReadOnly Property ShadowCopyFiles As Boolean

[JScript] public function get ShadowCopyFiles() : Boolean;

Description

Gets an indication whether all assemblies that are loaded in the application domain are shadow copied.

This method sets the **System.AppDomainSetup.ShadowCopyFiles** property of the internal **System.AppDomainSetup** object associated with this instance.

ToString

[C#] public event AssemblyLoadEventHandler AssemblyLoad;

[C++] public: __sealed __event AssemblyLoadEventHandler* AssemblyLoad;

[VB] NotOverridable Public Event AssemblyLoad As

AssemblyLoadEventHandler

Description

Occurs when an assembly is loaded.

The **System.AssemblyLoadEventHandler** for this event can attempt to locate the assembly and load it.

ToString

[C#] public event ResolveEventHandler AssemblyResolve;

[C++] public: __sealed __event ResolveEventHandler* AssemblyResolve;

[VB] NotOverridable Public Event AssemblyResolve As ResolveEventHandler

Description

Occurs when the resolution of an assembly fails.

The **System.ResolveEventHandler** for this event can attempt to locate the assembly and load it.

ToString

1
2 [C#] public event EventHandler DomainUnload;

3 [C++] public: __sealed __event EventHandler* DomainUnload;

4 [VB] NotOverridable Public Event DomainUnload As EventHandler

5
6 *Description*

7 Occurs when an **System.AppDomain** is about to be unloaded.

8 The **System.EventHandler** for this event can attempt to locate the
9 assembly and load it.

10 ToString

11
12 [C#] public event EventHandler ProcessExit;

13 [C++] public: __sealed __event EventHandler* ProcessExit;

14 [VB] NotOverridable Public Event ProcessExit As EventHandler

15
16 *Description*

17 Occurs when a process is about to exit.

18 The **System.EventHandler** for this event can perform termination
19 activities, such as closing files, releasing storage and so on, before the process
20 ends.

21 ToString

22
23 [C#] public event ResolveEventHandler ResourceResolve;

24 [C++] public: __sealed __event ResolveEventHandler* ResourceResolve;

25 [VB] NotOverridable Public Event ResourceResolve As ResolveEventHandler

1
2 *Description*

3 Occurs when the resolution of a resource fails.

4 The **System.ResolveEventHandler** for this event can attempt to locate the
5 resource and load it.

6 ToString

7
8 [C#] public event ResolveEventHandler TypeResolve;

9 [C++] public: __sealed __event ResolveEventHandler* TypeResolve;

10 [VB] NotOverridable Public Event TypeResolve As ResolveEventHandler

11
12 *Description*

13 Occurs when the resolution of a type fails.

14 The **System.ResolveEventHandler** for this event can attempt to locate the
15 type and load it.

16 ToString

17
18 [C#] public event UnhandledExceptionHandler UnhandledException;

19 [C++] public: __sealed __event UnhandledExceptionHandler*

20 UnhandledException;

21 [VB] NotOverridable Public Event UnhandledException As

22 UnhandledExceptionHandler

23
24 *Description*

25 Occurs when an exception is not caught by an event handler.

For more information about handling events, see .

AppendPrivatePath

[C#] public void AppendPrivatePath(string path);

[C++] public: __sealed void AppendPrivatePath(String* path);

[VB] NotOverridable Public Sub AppendPrivatePath(ByVal path As String)

[JScript] public function AppendPrivatePath(path : String);

Description

Appends the specified name of the directory to the private path.

The private path, or relative search path, is the path relative to the base directory where the assembly resolver probes for private assemblies. The name of the directory to be appended to the private path.

ClearPrivatePath

[C#] public void ClearPrivatePath();

[C++] public: __sealed void ClearPrivatePath();

[VB] NotOverridable Public Sub ClearPrivatePath()

[JScript] public function ClearPrivatePath();

Description

Resets the **System.AppDomainSetup.PrivateBinPath** for this instance to **null** .

ClearShadowCopyPath

```

1
2 [C#] public void ClearShadowCopyPath();
3 [C++] public: __sealed void ClearShadowCopyPath();
4 [VB] NotOverridable Public Sub ClearShadowCopyPath()
5 [JScript] public function ClearShadowCopyPath();
6

```

Description

Resets the **System.AppDomainSetup.ShadowCopyDirectories** property for this instance to **null**.

CreateComInstanceFrom

```

11
12 [C#] public ObjectHandle CreateComInstanceFrom(string assemblyName, string
13 typeName);
14 [C++] public: ObjectHandle* CreateComInstanceFrom(String* assemblyName,
15 String* typeName);
16 [VB] Public Function CreateComInstanceFrom(ByVal assemblyName As String,
17 ByVal typeName As String) As ObjectHandle
18 [JScript] public function CreateComInstanceFrom(assemblyName : String,
19 typeName : String) : ObjectHandle;
20

```

Description

Creates an instance of a COM object. Parameters specify the name of the assembly that can create the object and the name of the type of the object.

Return Value: An object that is a wrapper for the new instance.

1 Use this method to create types remotely without having to load the type
2 locally. The return value must to be unwrapped in order to access the real object.
3 The name of the assembly in which this object type resides. The type name of the
4 desired object.

5 CreateDomain

6
7 [C#] public static AppDomain CreateDomain(string friendlyName);
8 [C++] public: static AppDomain* CreateDomain(String* friendlyName);
9 [VB] Public Shared Function CreateDomain(ByVal friendlyName As String) As
10 AppDomain
11 [JScript] public static function CreateDomain(friendlyName : String) :
12 AppDomain;

13 14 *Description*

15 Creates a new application domain with the specified name.

16 *Return Value:* The newly created application domain.

17 *friendlyName* can be displayed in user interfaces to identify the domain.

18 The friendly name of the domain.

19 CreateDomain

20
21 [C#] public static AppDomain CreateDomain(string friendlyName, Evidence
22 securityInfo);
23 [C++] public: static AppDomain* CreateDomain(String* friendlyName,
24 Evidence* securityInfo);
25 [VB] Public Shared Function CreateDomain(ByVal friendlyName As String,

1 ByVal securityInfo As Evidence) As AppDomain

2 [JScript] public static function CreateDomain(friendlyName : String, securityInfo :
3 Evidence) : AppDomain; Creates a new application domain.

4
5 *Description*

6 Creates a new application domain with the given name using the supplied
7 evidence.

8 *Return Value:* The newly created application domain. The friendly name of the
9 domain. This friendly name can be displayed in user interfaces to identify the
10 domain. See the description of **System.AppDomain.FriendlyName**. Evidence
11 mapped through security policy to establish a top-of-stack permission set.

12 **CreateDomain**

13
14 [C#] public static AppDomain CreateDomain(string friendlyName, Evidence
15 securityInfo, AppDomainSetup info);

16 [C++] public: static AppDomain* CreateDomain(String* friendlyName,
17 Evidence* securityInfo, AppDomainSetup* info);

18 [VB] Public Shared Function CreateDomain(ByVal friendlyName As String,
19 ByVal securityInfo As Evidence, ByVal info As AppDomainSetup) As

20 AppDomain

21 [JScript] public static function CreateDomain(friendlyName : String, securityInfo :
22 Evidence, info : AppDomainSetup) : AppDomain;

23
24 *Description*

25

Creates a new application domain using the specified name, evidence, application domain setup information.

Return Value: The newly created application domain. The friendly name of the domain. This friendly name can be displayed in user interfaces to identify the domain. See the description of **System.AppDomain.FriendlyName**. Evidence mapped through security policy to establish a top-of-stack permission set. An object that contains application domain initialization information.

CreateDomain

```
[C#] public static AppDomain CreateDomain(string friendlyName, Evidence
securityInfo, string appBasePath, string appRelativeSearchPath, bool
shadowCopyFiles);
```

```
[C++] public: static AppDomain* CreateDomain(String* friendlyName,
Evidence* securityInfo, String* appBasePath, String* appRelativeSearchPath,
bool shadowCopyFiles);
```

```
[VB] Public Shared Function CreateDomain(ByVal friendlyName As String,
ByVal securityInfo As Evidence, ByVal appBasePath As String, ByVal
appRelativeSearchPath As String, ByVal shadowCopyFiles As Boolean) As
AppDomain
```

```
[JScript] public static function CreateDomain(friendlyName : String, securityInfo :
Evidence, appBasePath : String, appRelativeSearchPath : String,
shadowCopyFiles : Boolean) : AppDomain;
```

Description

Creates a new application domain with the given name using, evidence, application base path, relative search path, and a parameter that specifies whether a shadow copy of an assembly is to be loaded in to the application domain.

Return Value: The newly created application domain. The friendly name of the domain. This friendly name can be displayed in user interfaces to identify the domain. See the description of **System.AppDomain.FriendlyName**. Evidence mapped through security policy to establish a top-of-stack permission set. The base directory that the assembly resolver uses to probe for assemblies. See the description of **System.AppDomain.BaseDirectory**. The path relative to the base directory where the assembly resolver should probe for private assemblies. See the description of **System.AppDomain.RelativeSearchPath**. If **true**, a shadow copy of an assembly is loaded into this application domain.

CreateInstance

[C#] public ObjectHandle CreateInstance(string assemblyName, string typeName);

[C++] public: __sealed ObjectHandle* CreateInstance(String* assemblyName, String* typeName);

[VB] NotOverridable Public Function CreateInstance(ByVal assemblyName As String, ByVal typeName As String) As ObjectHandle

[JScript] public function CreateInstance(assemblyName : String, typeName : String) : ObjectHandle; Creates a new instance of a specified type defined in the specified assembly file.

Description

Creates a new instance of the specified type defined in the specified assembly.

Return Value: An object that is a wrapper for the new instance, or **null** if *typeName* is not found. The return value needs to be unwrapped to access the real object.

This is a convenience method that calls the default constructor for *typeName*. The display name of the assembly. See the description of **System.Reflection.AssemblyName** for the format of the display name. The full name of the type.

CreateInstance

```
[C#] public ObjectHandle CreateInstance(string assemblyName, string typeName, object[] activationAttributes);
```

```
[C++] public: __sealed ObjectHandle* CreateInstance(String* assemblyName, String* typeName, Object* activationAttributes __gc[]);
```

```
[VB] NotOverridable Public Function CreateInstance(ByVal assemblyName As String, ByVal typeName As String, ByVal activationAttributes() As Object) As ObjectHandle
```

```
[JScript] public function CreateInstance(assemblyName : String, typeName : String, activationAttributes : Object[]) : ObjectHandle;
```

Description

Creates an instance using the name of the type and the assembly where it exists.

Return Value: A handle to the requested object.

1 This method allows types to be created remotely without having to load the
2 type locally. This will return an **System.Runtime.Remoting.ObjectHandle** that
3 needs to be unwrapped in order to access the real object. The name of the
4 assembly in which this object type resides. The type name of the desired object.
5 One or more attributes that can participate in activation.

6 CreateInstance

7
8 [C#] public ObjectHandle CreateInstance(string assemblyName, string typeName,
9 bool ignoreCase, BindingFlags bindingAttr, Binder binder, object[] args,
10 CultureInfo culture, object[] activationAttributes, Evidence securityAttributes);
11 [C++] public: __sealed ObjectHandle* CreateInstance(String* assemblyName,
12 String* typeName, bool ignoreCase, BindingFlags bindingAttr, Binder* binder,
13 Object* args __gc[], CultureInfo* culture, Object* activationAttributes __gc[],
14 Evidence* securityAttributes);
15 [VB] NotOverridable Public Function CreateInstance(ByVal assemblyName As
16 String, ByVal typeName As String, ByVal ignoreCase As Boolean, ByVal
17 bindingAttr As BindingFlags, ByVal binder As Binder, ByVal args() As Object,
18 ByVal culture As CultureInfo, ByVal activationAttributes() As Object, ByVal
19 securityAttributes As Evidence) As ObjectHandle
20 [JScript] public function CreateInstance(assemblyName : String, typeName :
21 String, ignoreCase : Boolean, bindingAttr : BindingFlags, binder : Binder, args :
22 Object[], culture : CultureInfo, activationAttributes : Object[], securityAttributes :
23 Evidence) : ObjectHandle;

24 Description

Creates an instance using the name of the type and the assembly where it exists.

Return Value: A handle to the requested object.

This method allows types to be created remotely without having to load the type locally. This will return an **System.Runtime.Remoting.ObjectHandle** that needs to be unwrapped in order to access the real object. The name of the assembly in which this object type resides. The type name of the desired object. A Boolean value specifying whether to perform a case-sensitive search or not. This bitmask affects the way in which the search is conducted. The value is a combination of zero or more bit flags from **System.Reflection.BindingFlags**, such as **NonPublic** and **OABinding**. An object that enables the binding, coercion of argument types, invocation of members and retrieval of **System.Reflection.MemberInfo** objects using reflection. If *binder* is null, the default binder is used. The arguments to be passed to the constructor. This array of arguments must match in number, order, and type the parameters of the constructor to be invoked. If the default constructor is desired, *args* must be an empty array or null. An instance of **System.Globalization.CultureInfo** used to govern the coercion of types. If this is null, the **CultureInfo** for the current thread is used. (Note that this is necessary to, for example, convert a **String** that represents 1000 to a **Double** value, since 1000 is represented differently by different cultures.) One or more attributes that can participate in activation.

CreateInstanceAndUnwrap

[C#] public object CreateInstanceAndUnwrap(string assemblyName, string typeName);

1 [C++] public: Object* CreateInstanceAndUnwrap(String* assemblyName, String*
2 typeName);

3 [VB] Public Function CreateInstanceAndUnwrap(ByVal assemblyName As
4 String, ByVal typeName As String) As Object

5 [JScript] public function CreateInstanceAndUnwrap(assemblyName : String,
6 typeName : String) : Object; Creates a new instance of a specified type.

8 *Description*

9 Creates a new instance of the specified type. Parameters specify the
10 assembly where the type is defined, and the name of the type.

11 *Return Value:* An instance of *typeName* , or **null** if *typeName* is not found.

12 This a convenience method that combines
13 **System.AppDomain.CreateInstance(System.String,System.String)** and
14 **System.Runtime.Remoting.ObjectHandle.Unwrap** . This method calls the
15 default constructor for *typeName* . The name of the assembly. The fully qualified
16 name of the type.

17 CreateInstanceAndUnwrap

18
19 [C#] public object CreateInstanceAndUnwrap(string assemblyName, string
20 typeName, object[] activationAttributes);

21 [C++] public: Object* CreateInstanceAndUnwrap(String* assemblyName, String*
22 typeName, Object* activationAttributes __gc[]);

23 [VB] Public Function CreateInstanceAndUnwrap(ByVal assemblyName As
24 String, ByVal typeName As String, ByVal activationAttributes() As Object) As
25 Object

1 [JScript] public function CreateInstanceAndUnwrap(assemblyName : String,
2 typeName : String, activationAttributes : Object[]) : Object;

4 *Description*

5 Creates a new instance of the specified type. Parameters specify the
6 assembly where the type is defined, the name of the type, and an array of
7 activation attributes.

8 *Return Value:* An instance of *typeName* , or **null** if *typeName* is not found.

9 This a convenience method that combines
10 **System.AppDomain.CreateInstance(System.String,System.String)** and
11 **System.Runtime.Remoting.ObjectHandle.Unwrap** . This method calls the
12 default constructor for *typeName* . The name of the assembly. The fully qualified
13 name of the type. An array containing one or more attributes that can participate in
14 activation.

15 **CreateInstanceAndUnwrap**

16
17 [C#] public object CreateInstanceAndUnwrap(string assemblyName, string
18 typeName, bool ignoreCase, BindingFlags bindingAttr, Binder binder, object[]
19 args, CultureInfo culture, object[] activationAttributes, Evidence
20 securityAttributes);

21 [C++] public: Object* CreateInstanceAndUnwrap(String* assemblyName, String*
22 typeName, bool ignoreCase, BindingFlags bindingAttr, Binder* binder, Object*
23 args __gc[], CultureInfo* culture, Object* activationAttributes __gc[], Evidence*
24 securityAttributes);

25 [VB] Public Function CreateInstanceAndUnwrap(ByVal assemblyName As

```

1 String, ByVal typeName As String, ByVal ignoreCase As Boolean, ByVal
2 bindingAttr As BindingFlags, ByVal binder As Binder, ByVal args() As Object,
3 ByVal culture As CultureInfo, ByVal activationAttributes() As Object, ByVal
4 securityAttributes As Evidence) As Object
5 [JScript] public function CreateInstanceAndUnwrap(assemblyName : String,
6 typeName : String, ignoreCase : Boolean, bindingAttr : BindingFlags, binder :
7 Binder, args : Object[], culture : CultureInfo, activationAttributes : Object[],
8 securityAttributes : Evidence) : Object;

```

Description

Creates a new instance of the specified type. Parameters specify the name of the type and how it is found and created.

Return Value: An **System.Object** , which is an instance of *typeName* , or **null** if *typeName* is not found.

This a convenience method that combines **System.AppDomain.CreateInstance(System.String, System.String)** and **System.Runtime.Remoting.ObjectHandle.Unwrap** . The name of the assembly. The fully qualified name of the type. A Boolean value specifying whether to perform a case-sensitive search or not. This bitmask affects the way in which the search is conducted. The value is a combination of zero or more bit flags from **System.Reflection.BindingFlags**, such as **NonPublic** and **OABinding**. An object that enables the binding, coercion of argument types, invocation of members and retrieval of **System.Reflection.MemberInfo** objects using reflection. If *binder* is null, the default binder is used. The arguments to be passed to the constructor. This array of arguments must match in number, order, and type the parameters of

the constructor to be invoked. If the default constructor is desired, *args* must be an empty array or null. A culture-specific object used to govern the coercion of types. If *culture* is **null**, the **CultureInfo** for the current thread is used. (Note that this is necessary to, for example, convert a **String** that represents 1000 to a **Double** value, since 1000 is represented differently by different cultures.) An **System.Object** array containing one or more attributes that can participate in activation. An **System.Security.Policy.Evidence** object used to verify that *typeName* is allowed to be created.

CreateInstanceFrom

```
[C#] public ObjectHandle CreateInstanceFrom(string assemblyFile, string
typeName);
[C++] public: __sealed ObjectHandle* CreateInstanceFrom(String* assemblyFile,
String* typeName);
[VB] NotOverridable Public Function CreateInstanceFrom(ByVal assemblyFile
As String, ByVal typeName As String) As ObjectHandle
[JScript] public function CreateInstanceFrom(assemblyFile : String, typeName :
String) : ObjectHandle; Creates a new instance of a specified type defined in the
specified assembly file.
```

Description

Creates a new instance of the specified type defined in the specified assembly file.

Return Value: An object that is a wrapper for the new instance, or **null** if

1 *typeName* is not found. The return value needs to be unwrapped to access the real
2 object.

3 This is a convenience method that calls the default constructor for
4 *typeName* . The assembly file name. The full name of the type.

5 CreateInstanceFrom

6
7 [C#] public ObjectHandle CreateInstanceFrom(string assemblyFile, string
8 typeName, object[] activationAttributes);

9 [C++] public: __sealed ObjectHandle* CreateInstanceFrom(String* assemblyFile,
10 String* typeName, Object* activationAttributes __gc[]);

11 [VB] NotOverridable Public Function CreateInstanceFrom(ByVal assemblyFile
12 As String, ByVal typeName As String, ByVal activationAttributes() As Object)
13 As ObjectHandle

14 [JScript] public function CreateInstanceFrom(assemblyFile : String, typeName :
15 String, activationAttributes : Object[]) : ObjectHandle;

17 *Description*

18 Creates a new instance of the specified type defined in the specified
19 assembly file.

20 *Return Value:* A handle to the requested object.

21 This method allows types to be created remotely without having to load the
22 type locally. This will return an **System.Runtime.Remoting.ObjectHandle** that
23 needs to be unwrapped in order to access the real object. The file containing the
24 desired object's assembly. The type name of the desired object. One or more
25 attributes that can participate in activation.

CreateInstanceFrom

```
[C#] public ObjectHandle CreateInstanceFrom(string assemblyFile, string
typeName, bool ignoreCase, BindingFlags bindingAttr, Binder binder, object[]
args, CultureInfo culture, object[] activationAttributes, Evidence
securityAttributes);

[C++] public: __sealed ObjectHandle* CreateInstanceFrom(String* assemblyFile,
String* typeName, bool ignoreCase, BindingFlags bindingAttr, Binder* binder,
Object* args __gc[], CultureInfo* culture, Object* activationAttributes __gc[],
Evidence* securityAttributes);

[VB] NotOverridable Public Function CreateInstanceFrom(ByVal assemblyFile
As String, ByVal typeName As String, ByVal ignoreCase As Boolean, ByVal
bindingAttr As BindingFlags, ByVal binder As Binder, ByVal args() As Object,
ByVal culture As CultureInfo, ByVal activationAttributes() As Object, ByVal
securityAttributes As Evidence) As ObjectHandle

[JScript] public function CreateInstanceFrom(assemblyFile : String, typeName :
String, ignoreCase : Boolean, bindingAttr : BindingFlags, binder : Binder, args :
Object[], culture : CultureInfo, activationAttributes : Object[], securityAttributes :
Evidence) : ObjectHandle;
```

Description

Creates a new instance of the specified type defined in the specified assembly file.

Return Value: A handle to the requested object.

This method allows types to be created remotely without having to load the type locally. This will return an **System.Runtime.Remoting.ObjectHandle** that needs to be unwrapped in order to access the real object. The file for the assembly in which this object type resides. The type name of the desired object. A Boolean value specifying whether to perform a case-sensitive search or not. This bitmask affects the way in which the search is conducted. The value is a combination of zero or more bit flags from **System.Reflection.BindingFlags**, such as **NonPublic** and **OABinding**. An object that enables the binding, coercion of argument types, invocation of members and retrieval of **System.Reflection.MemberInfo** objects through reflection. If *binder* is null, the default binder is used. The arguments to be passed to the constructor. This array of arguments must match in number, order, and type the parameters of the constructor to be invoked. If the default constructor is desired, *args* must be an empty array or null. An instance of **System.Globalization.CultureInfo** used to govern the coercion of types. If this is null, the **CultureInfo** for the current thread is used. (Note that this is necessary to, for example, convert a **String** that represents 1000 to a **Double** value, since 1000 is represented differently by different cultures.) One or more attributes that can participate in activation.

CreateInstanceFromAndUnwrap

[C#] public object CreateInstanceFromAndUnwrap(string assemblyName, string typeName);

[C++] public: Object* CreateInstanceFromAndUnwrap(String* assemblyName, String* typeName);

[VB] Public Function CreateInstanceFromAndUnwrap(ByVal assemblyName As

String, ByVal typeName As String) As Object

[JScript] public function CreateInstanceFromAndUnwrap(assemblyName : String, typeName : String) : Object; Creates a new instance of a specified type defined in the specified assembly file.

Description

Creates a new instance of the specified type defined in the specified assembly file.

Return Value: Returns an **System.Runtime.Remoting.ObjectHandle** object that is a wrapper for the new instance. Returns **null** if the type is not found.

This a convenience method that combines **System.AppDomain.CreateInstance(System.String, System.String)** and **System.Runtime.Remoting.ObjectHandle.Unwrap** . This method calls the default constructor for *typeName* . The assembly file name. The full name of the type.

CreateInstanceFromAndUnwrap

[C#] public object CreateInstanceFromAndUnwrap(string assemblyName, string typeName, object[] activationAttributes);

[C++] public: Object* CreateInstanceFromAndUnwrap(String* assemblyName, String* typeName, Object* activationAttributes __gc[]);

[VB] Public Function CreateInstanceFromAndUnwrap(ByVal assemblyName As String, ByVal typeName As String, ByVal activationAttributes() As Object) As Object

[JScript] public function CreateInstanceFromAndUnwrap(assemblyName : String,

1 typeName : String, activationAttributes : Object[]) : Object;

3 *Description*

4 Creates an instance using the name of the type and the assembly where it
5 exists. This allows types to be created remotely without having to load the type
6 locally. This will return an ObjectHandle that needs to be unwrapped in order to
7 access the real object.

8 *Return Value:* A handle to the desired object.

9 This a convenience method that combines
10 **System.AppDomain.CreateInstance(System.String, System.String)** and
11 **System.Runtime.Remoting.ObjectHandle.Unwrap** . This method calls the
12 default constructor for *typeName* . The file containing the desired object's
13 assembly. The type name of the desired object. One or more attributes that can
14 participate in activation.

15 **CreateInstanceFromAndUnwrap**

16
17 [C#] public object CreateInstanceFromAndUnwrap(string assemblyName, string
18 typeName, bool ignoreCase, BindingFlags bindingAttr, Binder binder, object[]
19 args, CultureInfo culture, object[] activationAttributes, Evidence
20 securityAttributes);

21 [C++] public: Object* CreateInstanceFromAndUnwrap(String* assemblyName,
22 String* typeName, bool ignoreCase, BindingFlags bindingAttr, Binder* binder,
23 Object* args __gc[], CultureInfo* culture, Object* activationAttributes __gc[],
24 Evidence* securityAttributes);

25 [VB] Public Function CreateInstanceFromAndUnwrap(ByVal assemblyName As

```

1 String, ByVal typeName As String, ByVal ignoreCase As Boolean, ByVal
2 bindingAttr As BindingFlags, ByVal binder As Binder, ByVal args() As Object,
3 ByVal culture As CultureInfo, ByVal activationAttributes() As Object, ByVal
4 securityAttributes As Evidence) As Object
5 [JScript] public function CreateInstanceFromAndUnwrap(assemblyName : String,
6 typeName : String, ignoreCase : Boolean, bindingAttr : BindingFlags, binder :
7 Binder, args : Object[], culture : CultureInfo, activationAttributes : Object[],
8 securityAttributes : Evidence) : Object;

```

Description

Creates an instance using the name of the type and the assembly where it exists. This allows types to be created remotely without having to load the type locally. This will return an `ObjectHandle` that needs to be unwrapped in order to access the real object.

Return Value: A handle to the requested object.

This a convenience method that combines **System.AppDomain.CreateInstance(System.String, System.String)** and **System.Runtime.Remoting.ObjectHandle.Unwrap**. The file for the assembly in which this object type resides. The type name of the desired object. A Boolean value specifying whether to perform a case-sensitive search or not. This bitmask affects the way in which the search is conducted. The value is a combination of zero or more bit flags from **System.Reflection.BindingFlags**, such as **NonPublic** and **OABinding**. An object that enables the binding, coercion of argument types, invocation of members and retrieval of **System.Reflection.MemberInfo** objects through reflection. If *binder* is null, the

1 default binder is used. The arguments to be passed to the constructor. This array of
2 arguments must match in number, order, and type the parameters of the
3 constructor to be invoked. If the default constructor is desired, *args* must be an
4 empty array or null. An instance of **System.Globalization.CultureInfo** used to
5 govern the coercion of types. If *culture* is **null**, the **CultureInfo** for the current
6 thread is used. (Note that this is necessary to, for example, convert a **String** that
7 represents 1000 to a **Double** value, since 1000 is represented differently by
8 different cultures.) One or more attributes that can participate in activation.

9 DefineDynamicAssembly

10
11 [C#] public AssemblyBuilder DefineDynamicAssembly(AssemblyName name,
12 AssemblyBuilderAccess access);

13 [C++] public: __sealed AssemblyBuilder*
14 DefineDynamicAssembly(AssemblyName* name, AssemblyBuilderAccess
15 access);

16 [VB] NotOverridable Public Function DefineDynamicAssembly(ByVal name As
17 AssemblyName, ByVal access As AssemblyBuilderAccess) As AssemblyBuilder

18 [JScript] public function DefineDynamicAssembly(name : AssemblyName, access
19 : AssemblyBuilderAccess) : AssemblyBuilder; Defines a dynamic assembly in the
20 current application domain.

21 Description

22 Defines a dynamic assembly with the given name and the given access.

23
24 *Return Value:* Represents the dynamic assembly created.

1 You can specify partial signing of the assembly by specifying
2 `AssemblyName.Originator`. You can specify full signing of the assembly by
3 specifying `AssemblyName.Originator` and `AssemblyName.KeyPair`. The "strong
4 name", or unique identity of the dynamic assembly. The access mode for the
5 dynamic assembly.

6 `DefineDynamicAssembly`

7
8 [C#] `public AssemblyBuilder DefineDynamicAssembly(AssemblyName name,`
9 `AssemblyBuilderAccess access, Evidence evidence);`

10 [C++] `public: __sealed AssemblyBuilder*`

11 `DefineDynamicAssembly(AssemblyName* name, AssemblyBuilderAccess`
12 `access, Evidence* evidence);`

13 [VB] `NotOverridable Public Function DefineDynamicAssembly(ByVal name As`
14 `AssemblyName, ByVal access As AssemblyBuilderAccess, ByVal evidence As`
15 `Evidence) As AssemblyBuilder`

16 [JScript] `public function DefineDynamicAssembly(name : AssemblyName, access`
17 `: AssemblyBuilderAccess, evidence : Evidence) : AssemblyBuilder;`

18 19 *Description*

20 Defines a dynamic assembly with the given name, the given access, and the
21 supplied evidence.

22 *Return Value:* Represents the dynamic assembly created.

23 Only fully trusted callers can supply evidence when defining a dynamic
24 assembly. The runtime will map the supplied evidence through policy to determine
25 the granted permissions. Semi-trusted callers must supply a **null** evidence. If

evidence is **null** , the runtime copies the permission sets, that is, the current grant and deny sets, from the caller's assembly to the dynamic assembly being defined and marks policy as resolved. If the dynamic assembly is saved to disk, subsequent loads will get grants based on policies associated with the location where the assembly was saved. The unique identity of the dynamic assembly. The mode in which the dynamic assembly will be accessed. The evidence supplied for the dynamic assembly.

DefineDynamicAssembly

```
[C#] public AssemblyBuilder DefineDynamicAssembly(AssemblyName name,
AssemblyBuilderAccess access, string dir);
[C++] public: __sealed AssemblyBuilder*
DefineDynamicAssembly(AssemblyName* name, AssemblyBuilderAccess
access, String* dir);
[VB] NotOverridable Public Function DefineDynamicAssembly(ByVal name As
AssemblyName, ByVal access As AssemblyBuilderAccess, ByVal dir As String)
As AssemblyBuilder
[JScript] public function DefineDynamicAssembly(name : AssemblyName, access
: AssemblyBuilderAccess, dir : String) : AssemblyBuilder;
```

Description

Defines a dynamic assembly with the given name, the given access, and the name of the directory for saving the assembly.

Return Value: Represents the dynamic assembly created.

1 You can specify partial signing of the assembly by specifying
2 `AssemblyName.Originator`. You can specify full signing of the assembly by
3 specifying `AssemblyName.Originator` and `AssemblyName.KeyPair`. The unique
4 identity of the dynamic assembly. The mode in which the dynamic assembly will
5 be accessed. The name of the directory in which the assembly will be saved. If *dir*
6 is **null** , the directory defaults to the current directory.

7 DefineDynamicAssembly

8
9 [C#] public AssemblyBuilder DefineDynamicAssembly(AssemblyName name,
10 AssemblyBuilderAccess access, string dir, Evidence evidence);
11 [C++] public: __sealed AssemblyBuilder*
12 DefineDynamicAssembly(AssemblyName* name, AssemblyBuilderAccess
13 access, String* dir, Evidence* evidence);
14 [VB] NotOverridable Public Function DefineDynamicAssembly(ByVal name As
15 AssemblyName, ByVal access As AssemblyBuilderAccess, ByVal dir As String,
16 ByVal evidence As Evidence) As AssemblyBuilder
17 [JScript] public function DefineDynamicAssembly(name : AssemblyName, access
18 : AssemblyBuilderAccess, dir : String, evidence : Evidence) : AssemblyBuilder;

19 Description

20
21 Defines a dynamic assembly with the given name, given access, the name
22 of the directory for saving the assembly, and the supplied evidence.

23 *Return Value:* Represents the dynamic assembly created.

24 Only fully trusted callers can supply their *evidence* when defining a
25 dynamic assembly. The runtime will map the evidence through policy to

determine the granted permissions. Semi-trusted callers must supply a **null** evidence. If *evidence* is **null**, the runtime copies the permission sets, that is, the current grant and deny sets, from the caller's assembly to the dynamic assembly being defined and marks policy as resolved. If the dynamic assembly is saved to disk, subsequent loads will get grants based on policies associated with the location where the assembly was saved. The unique identity of the dynamic assembly. The mode in which the dynamic assembly will be accessed. The name of the directory in which the assembly will be saved. If *dir* is **null**, the directory defaults to the current directory. The evidence supplied for the dynamic assembly.

DefineDynamicAssembly

```
[C#] public AssemblyBuilder DefineDynamicAssembly(AssemblyName name,
AssemblyBuilderAccess access, PermissionSet requiredPermissions,
PermissionSet optionalPermissions, PermissionSet refusedPermissions);

[C++] public: __sealed AssemblyBuilder*
DefineDynamicAssembly(AssemblyName* name, AssemblyBuilderAccess
access, PermissionSet* requiredPermissions, PermissionSet* optionalPermissions,
PermissionSet* refusedPermissions);

[VB] NotOverridable Public Function DefineDynamicAssembly(ByVal name As
AssemblyName, ByVal access As AssemblyBuilderAccess, ByVal
requiredPermissions As PermissionSet, ByVal optionalPermissions As
PermissionSet, ByVal refusedPermissions As PermissionSet) As AssemblyBuilder

[JScript] public function DefineDynamicAssembly(name : AssemblyName, access
: AssemblyBuilderAccess, requiredPermissions : PermissionSet,
optionalPermissions : PermissionSet, refusedPermissions : PermissionSet) :
```


1 AssemblyBuilder;

3 *Description*

4 Defines a dynamic assembly with the given name, the given access, and the
5 given permission requests.

6 *Return Value:* Represents the dynamic assembly created.

7 You can specify partial signing of the assembly by specifying
8 AssemblyName.Originator. You can specify full signing of the assembly by
9 specifying AssemblyName.Originator and AssemblyName.KeyPair. The unique
10 identity of the dynamic assembly. The mode in which the dynamic assembly will
11 be accessed. The required permission request. The optional permission request.
12 The refused permission request.

13 DefineDynamicAssembly

14
15 [C#] public AssemblyBuilder DefineDynamicAssembly(AssemblyName name,
16 AssemblyBuilderAccess access, Evidence evidence, PermissionSet
17 requiredPermissions, PermissionSet optionalPermissions, PermissionSet
18 refusedPermissions);

19 [C++] public: __sealed AssemblyBuilder*
20 DefineDynamicAssembly(AssemblyName* name, AssemblyBuilderAccess
21 access, Evidence* evidence, PermissionSet* requiredPermissions, PermissionSet*
22 optionalPermissions, PermissionSet* refusedPermissions);

23 [VB] NotOverridable Public Function DefineDynamicAssembly(ByVal name As
24 AssemblyName, ByVal access As AssemblyBuilderAccess, ByVal evidence As
25 Evidence, ByVal requiredPermissions As PermissionSet, ByVal

```

1 optionalPermissions As PermissionSet, ByVal refusedPermissions As
2 PermissionSet) As AssemblyBuilder
3 [JScript] public function DefineDynamicAssembly(name : AssemblyName, access
4 : AssemblyBuilderAccess, evidence : Evidence, requiredPermissions :
5 PermissionSet, optionalPermissions : PermissionSet, refusedPermissions :
6 PermissionSet) : AssemblyBuilder;

```

8 *Description*

9 Defines a dynamic assembly with the given name, given access, supplied
10 evidence, and the permission requests.

11 *Return Value:* Represents the dynamic assembly created.

12 Only fully trusted callers can supply their *evidence* when defining a
13 dynamic assembly. The runtime will map the evidence through policy to
14 determine the granted permissions. Semi-trusted callers must supply a **null**
15 evidence. If *evidence* is **null**, the runtime copies the permission sets (that is, the
16 current grant and deny sets) from the caller's assembly to the dynamic assembly
17 being defined and marks policy as resolved. The unique identity of the dynamic
18 assembly. The mode in which the dynamic assembly will be accessed. The
19 evidence supplied for the dynamic assembly. The required permission request. The
20 optional permission request. The refused permission request.

21 DefineDynamicAssembly

```

22
23 [C#] public AssemblyBuilder DefineDynamicAssembly(AssemblyName name,
24 AssemblyBuilderAccess access, string dir, PermissionSet requiredPermissions,
25 PermissionSet optionalPermissions, PermissionSet refusedPermissions);

```

```

1  [C++] public: __sealed AssemblyBuilder*
2  DefineDynamicAssembly(AssemblyName* name, AssemblyBuilderAccess
3  access, String* dir, PermissionSet* requiredPermissions, PermissionSet*
4  optionalPermissions, PermissionSet* refusedPermissions);
5  [VB] NotOverridable Public Function DefineDynamicAssembly(ByVal name As
6  AssemblyName, ByVal access As AssemblyBuilderAccess, ByVal dir As String,
7  ByVal requiredPermissions As PermissionSet, ByVal optionalPermissions As
8  PermissionSet, ByVal refusedPermissions As PermissionSet) As AssemblyBuilder
9  [JScript] public function DefineDynamicAssembly(name : AssemblyName, access
10 : AssemblyBuilderAccess, dir : String, requiredPermissions : PermissionSet,
11 optionalPermissions : PermissionSet, refusedPermissions : PermissionSet) :
12 AssemblyBuilder;

```

Description

Defines a dynamic assembly with the given name, given access, the name of the directory for saving the assembly, and the permission requests.

Return Value: Represents the dynamic assembly created.

You can specify partial signing of the assembly by specifying `AssemblyName.Originator`. You can specify full signing of the assembly by specifying `AssemblyName.Originator` and `AssemblyName.KeyPair`. The unique identity of the dynamic assembly. The mode in which the dynamic assembly will be accessed. The name of the directory in which the assembly will be saved. If *dir* is **null**, the directory defaults to the current directory. The required permission request. The optional permission request. The refused permission request.

`DefineDynamicAssembly`

```

1
2 [C#] public AssemblyBuilder DefineDynamicAssembly(AssemblyName name,
3 AssemblyBuilderAccess access, string dir, Evidence evidence, PermissionSet
4 requiredPermissions, PermissionSet optionalPermissions, PermissionSet
5 refusedPermissions);
6 [C++] public: __sealed AssemblyBuilder*
7 DefineDynamicAssembly(AssemblyName* name, AssemblyBuilderAccess
8 access, String* dir, Evidence* evidence, PermissionSet* requiredPermissions,
9 PermissionSet* optionalPermissions, PermissionSet* refusedPermissions);
10 [VB] NotOverridable Public Function DefineDynamicAssembly(ByVal name As
11 AssemblyName, ByVal access As AssemblyBuilderAccess, ByVal dir As String,
12 ByVal evidence As Evidence, ByVal requiredPermissions As PermissionSet,
13 ByVal optionalPermissions As PermissionSet, ByVal refusedPermissions As
14 PermissionSet) As AssemblyBuilder
15 [JScript] public function DefineDynamicAssembly(name : AssemblyName, access
16 : AssemblyBuilderAccess, dir : String, evidence : Evidence, requiredPermissions :
17 PermissionSet, optionalPermissions : PermissionSet, refusedPermissions :
18 PermissionSet) : AssemblyBuilder;
19

```

Description

Defines a dynamic assembly with the given name, given access, the name of the directory for saving the assembly, supplied evidence, and the permission requests.

Return Value: Represents the dynamic assembly created.

1 Only fully trusted callers can supply their *evidence* when defining a
2 dynamic assembly. The runtime will map the evidence through policy to
3 determine the granted permissions. Semi-trusted callers must supply a **null**
4 evidence. If *evidence* is **null**, the runtime copies the permission sets, that is, the
5 current grant and deny sets, from the caller's assembly to the dynamic assembly
6 being defined and marks policy as resolved. The unique identity of the dynamic
7 assembly. The mode in which the dynamic assembly will be accessed. The name
8 of the directory in which the assembly will be saved. If *dir* is **null**, the directory
9 defaults to the current directory. The evidence supplied for the dynamic assembly.
10 The required permission request. The optional permission request. The refused
11 permission request.

12 DefineDynamicAssembly

13
14 [C#] public AssemblyBuilder DefineDynamicAssembly(AssemblyName name,
15 AssemblyBuilderAccess access, string dir, Evidence evidence, PermissionSet
16 requiredPermissions, PermissionSet optionalPermissions, PermissionSet
17 refusedPermissions, bool isSynchronized);
18 [C++] public: __sealed AssemblyBuilder*
19 DefineDynamicAssembly(AssemblyName* name, AssemblyBuilderAccess
20 access, String* dir, Evidence* evidence, PermissionSet* requiredPermissions,
21 PermissionSet* optionalPermissions, PermissionSet* refusedPermissions, bool
22 isSynchronized);
23 [VB] NotOverridable Public Function DefineDynamicAssembly(ByVal name As
24 AssemblyName, ByVal access As AssemblyBuilderAccess, ByVal dir As String,
25 ByVal evidence As Evidence, ByVal requiredPermissions As PermissionSet,

ByVal optionalPermissions As PermissionSet, ByVal refusedPermissions As
 PermissionSet, ByVal isSynchronized As Boolean) As AssemblyBuilder
 [JScript] public function DefineDynamicAssembly(name : AssemblyName, access
 : AssemblyBuilderAccess, dir : String, evidence : Evidence, requiredPermissions :
 PermissionSet, optionalPermissions : PermissionSet, refusedPermissions :
 PermissionSet, isSynchronized : Boolean) : AssemblyBuilder;

Description

Defines a dynamic assembly using the specified name, access mode,
 storage directory, evidence, permission requests, and synchronization option.

Return Value: Represents the dynamic assembly created.

Only fully trusted callers can supply their *evidence* when defining a
 dynamic **System.Reflection.Assembly** . The runtime will map the
System.Security.Policy.Evidence through policy to determine the granted
 permissions. Semi-trusted callers must supply an
System.Security.Policy.Evidence . If *evidence* is **null** , the runtime copies the
 permission sets, that is, the current grant and deny sets, from the caller's
System.Reflection.Assembly to the dynamic **System.Reflection.Assembly** being
 defined and marks policy as resolved. The unique identity of the dynamic
 assembly. The mode in which the dynamic assembly will be accessed. The name
 of the directory in which the dynamic assembly will be saved. If *dir* is **null** , the
 directory defaults to the current directory. The evidence supplied for the dynamic
 assembly. The required permission request. The optional permission request. The
 refused permission request. If **true** , the creation of modules, types, and members
 in the dynamic assembly are synchronized.

DoCallBack

[C#] public void DoCallBack(CrossAppDomainDelegate callBackDelegate);

[C++] public: __sealed void DoCallBack(CrossAppDomainDelegate*
callBackDelegate);

[VB] NotOverridable Public Sub DoCallBack(ByVal callBackDelegate As
CrossAppDomainDelegate)

[JScript] public function DoCallBack(callBackDelegate :
CrossAppDomainDelegate);

Description

Executes the code in another application domain that is identified by the
specified delegate.

callBackDelegate can specify a marshal-by-value,
System.MarshalByRefObject, or **System.ContextBoundObject** object. A
delegate that specifies a method to call.

ExecuteAssembly

[C#] public int ExecuteAssembly(string assemblyFile);

[C++] public: __sealed int ExecuteAssembly(String* assemblyFile);

[VB] NotOverridable Public Function ExecuteAssembly(ByVal assemblyFile As
String) As Integer

[JScript] public function ExecuteAssembly(assemblyFile : String) : int;

Description

Execute the **System.Reflection.Assembly** given its file name. The method specified in the .NET Framework header is called.

Return Value: The value returned by the entry point of the assembly.

The method does not spawn a new process, create a new application domain, or execute the entry point method on a new thread. The name of the file from which the assembly is to be loaded.

ExecuteAssembly

```
[C#] public int ExecuteAssembly(string assemblyFile, Evidence  
assemblySecurity);
```

```
[C++] public: __sealed int ExecuteAssembly(String* assemblyFile, Evidence*  
assemblySecurity);
```

```
[VB] NotOverridable Public Function ExecuteAssembly(ByVal assemblyFile As  
String, ByVal assemblySecurity As Evidence) As Integer
```

```
[JScript] public function ExecuteAssembly(assemblyFile : String,  
assemblySecurity : Evidence) : int; Executes the specified assembly.
```

Description

Execute the **System.Reflection.Assembly** given its file name and supplied evidence.

Return Value: The value returned by the entry point of the assembly.

The method does not spawn a new process, create a new application domain, or execute the entry point method on a new thread. The name of the file from which the assembly is to be loaded. Evidence for loading the assembly.

ExecuteAssembly


```

1
2 [C#] public int ExecuteAssembly(string assemblyFile, Evidence
3 assemblySecurity, string[] args);
4 [C++] public: __sealed int ExecuteAssembly(String* assemblyFile, Evidence*
5 assemblySecurity, String* args __gc[]);
6 [VB] NotOverridable Public Function ExecuteAssembly(ByVal assemblyFile As
7 String, ByVal assemblySecurity As Evidence, ByVal args() As String) As Integer
8 [JScript] public function ExecuteAssembly(assemblyFile : String,
9 assemblySecurity : Evidence, args : String[]) : int; Executes the assembly given its
10 file name.
11

```

Description

Execute the **System.Reflection.Assembly** given its file name and supplied **System.Security.Policy.Evidence** . Optionally, the **System.Reflection.Assembly** can be loaded into the domain-neutral code area for use by multiple AppDomains.

Return Value: The value returned by the entry point of the assembly.

The method does not spawn a new process, create a new application domain, or execute the entry point method on a new thread. The name of the file from which the assembly is to be loaded. The supplied evidence for the assembly. The arguments to the entry point of the assembly.

GetAssemblies

```

21
22
23 [C#] public Assembly[] GetAssemblies();
24 [C++] public: __sealed Assembly* GetAssemblies() [];
25 [VB] NotOverridable Public Function GetAssemblies() As Assembly()

```

1 [JScript] public function GetAssemblies() : Assembly[];

3 *Description*

4 Gets the assemblies that have been loaded into this application domain.

5 *Return Value:* An array of assemblies in this application domain.

6 **GetCurrentThreadId**

8 [C#] public static int GetCurrentThreadId();

9 [C++] public: static int GetCurrentThreadId();

10 [VB] Public Shared Function GetCurrentThreadId() As Integer

11 [JScript] public static function GetCurrentThreadId() : int;

13 *Description*

14 Gets the current thread identifier.

15 *Return Value:* A 32-bit signed integer that is the identifier of the current thread.

16 **GetData**

18 [C#] public object GetData(string name);

19 [C++] public: __sealed Object* GetData(String* name);

20 [VB] NotOverridable Public Function GetData(ByVal name As String) As Object

21 [JScript] public function GetData(name : String) : Object;

23 *Description*

Gets the value stored in the current application domain for the specified data name.

Return Value: The value of the *name* property.

name can be the value of one of the **System.AppDomainSetup** properties.

The name of an application domain property.

GetType

[C#] public Type GetType();

[C++] public: __sealed Type* GetType();

[VB] NotOverridable Public Function GetType() As Type

[JScript] public function GetType() : Type;

Description

Gets the type of the current instance.

Return Value: A **System.Type** object.

Description

Gets the type of the current instance.

Return Value: A **System.Type** object.

InitializeLifetimeService

[C#] public override object InitializeLifetimeService();

[C++] public: Object* InitializeLifetimeService();

[VB] Overrides Public Function InitializeLifetimeService() As Object

[JScript] public override function InitializeLifetimeService() : Object;

Description

Gives the **System.AppDomain** an infinite lifetime by preventing a lease from being created.

Return Value: Always **null**.

IsFinalizingForUnload

[C#] public bool IsFinalizingForUnload();

[C++] public: bool IsFinalizingForUnload();

[VB] Public Function IsFinalizingForUnload() As Boolean

[JScript] public function IsFinalizingForUnload() : Boolean;

Description

Indicates whether the common language runtime has started forcing objects to finalize.

Return Value: **true** if the common language runtime has started invoking finalizers, forcing objects to finalize; otherwise, **false**.

Some of the **System.AppDomain** infrastructure might have been garbage collected before the finalizers started running.

Load

[C#] public Assembly Load(AssemblyName assemblyRef);

[C++] public: __sealed Assembly* Load(AssemblyName* assemblyRef);

[VB] NotOverridable Public Function Load(ByVal assemblyRef As

AssemblyName) As Assembly

1 [JScript] public function Load(assemblyRef : AssemblyName) : Assembly; Loads
2 an **System.Reflection.Assembly** into this application domain.

3
4 *Description*

5 Loads an **System.Reflection.Assembly** given its
6 **System.Reflection.AssemblyName** .

7 *Return Value:* The loaded assembly.

8 This method should only be used to load an assembly into the current
9 application domain. This method is defined for interoperability callers who cannot
10 call the static Assembly.Load method. An object that describes the assembly to be
11 loaded.

12 Load

13
14 [C#] public Assembly Load(byte[] rawAssembly);

15 [C++] public: __sealed Assembly* Load(unsigned char rawAssembly __gc[]);

16 [VB] NotOverridable Public Function Load(ByVal rawAssembly() As Byte) As
17 Assembly

18 [JScript] public function Load(rawAssembly : Byte[]) : Assembly;

19
20 *Description*

21 Loads the **System.Reflection.Assembly** with a COFF based image
22 containing an emitted **System.Reflection.Assembly** .

23 *Return Value:* The loaded assembly.

24 This method should only be used to load an assembly into the current
25 application domain. This method is defined for interoperability callers who cannot

call the static `Assembly.Load` method. An array of type **byte** that is a COFF-based image containing an emitted assembly.

Load

[C#] `public Assembly Load(string assemblyString);`

[C++] `public: __sealed Assembly* Load(String* assemblyString);`

[VB] `NotOverridable Public Function Load(ByVal assemblyString As String) As Assembly`

[JScript] `public function Load(assemblyString : String) : Assembly;`

Description

Loads an **System.Reflection.Assembly** given its display name.

Return Value: The loaded assembly.

This method should only be used to load an assembly into the current application domain. This method is defined for interoperability callers who cannot call the static `Assembly.Load` method. The display name of the assembly.

Load

[C#] `public Assembly Load(AssemblyName assemblyRef, Evidence assemblySecurity);`

[C++] `public: __sealed Assembly* Load(AssemblyName* assemblyRef, Evidence* assemblySecurity);`

[VB] `NotOverridable Public Function Load(ByVal assemblyRef As AssemblyName, ByVal assemblySecurity As Evidence) As Assembly`

[JScript] `public function Load(assemblyRef : AssemblyName, assemblySecurity :`

Evidence) : Assembly; Loads an **System.Reflection.Assembly** into this application domain.

Description

Loads an **System.Reflection.Assembly** given its **System.Reflection.AssemblyName** .

Return Value: The loaded assembly. An object that describes the assembly to be loaded. Evidence for loading the assembly.

Load

```
[C#] public Assembly Load(byte[] rawAssembly, byte[] rawSymbolStore);  
[C++] public: __sealed Assembly* Load(unsigned char rawAssembly __gc[],  
unsigned char rawSymbolStore __gc[]);  
[VB] NotOverridable Public Function Load(ByVal rawAssembly() As Byte,  
ByVal rawSymbolStore() As Byte) As Assembly  
[JScript] public function Load(rawAssembly : Byte[], rawSymbolStore : Byte[]) :  
Assembly;
```

Description

Loads the **System.Reflection.Assembly** with a COFF based image containing an emitted **System.Reflection.Assembly** . The raw bytes representing the symbols for the **System.Reflection.Assembly** are also loaded.

Return Value: The loaded assembly.

This method should only be used to load an assembly into the current application domain. This method is defined for interoperability callers who cannot

call the static `Assembly.Load` method. An array of type **byte** that is a COFF-based image containing an emitted assembly. An array of type **byte** containing the raw bytes representing the symbols for the assembly.

Load

[C#] public Assembly Load(string assemblyString, Evidence assemblySecurity);

[C++] public: __sealed Assembly* Load(String* assemblyString, Evidence* assemblySecurity);

[VB] NotOverridable Public Function Load(ByVal assemblyString As String, ByVal assemblySecurity As Evidence) As Assembly

[JScript] public function Load(assemblyString : String, assemblySecurity : Evidence) : Assembly;

Description

Loads an **System.Reflection.Assembly** given its display name.

Return Value: The loaded assembly. The display name of the assembly. Evidence for loading the assembly.

Load

[C#] public Assembly Load(byte[] rawAssembly, byte[] rawSymbolStore, Evidence securityEvidence);

[C++] public: __sealed Assembly* Load(unsigned char rawAssembly __gc[], unsigned char rawSymbolStore __gc[], Evidence* securityEvidence);

[VB] NotOverridable Public Function Load(ByVal rawAssembly() As Byte, ByVal rawSymbolStore() As Byte, ByVal securityEvidence As Evidence) As

Assembly

```
[JScript] public function Load(rawAssembly : Byte[], rawSymbolStore : Byte[],  
securityEvidence : Evidence) : Assembly;
```

Description

Loads the **System.Reflection.Assembly** with a COFF based image containing an emitted **System.Reflection.Assembly** . The raw bytes representing the symbols for the **System.Reflection.Assembly** are also loaded.

Return Value: The loaded assembly.

This method should only be used to load an assembly into the current application domain. This method is defined for interoperability callers who cannot call the static Assembly.Load method. An array of type **byte** that is a COFF-based image containing an emitted assembly. An array of type **byte** containing the raw bytes representing the symbols for the assembly. Evidence for loading the assembly.

SetAppDomainPolicy

```
[C#] public void SetAppDomainPolicy(PolicyLevel domainPolicy);  
[C++] public: __sealed void SetAppDomainPolicy(PolicyLevel* domainPolicy);  
[VB] NotOverridable Public Sub SetAppDomainPolicy(ByVal domainPolicy As  
PolicyLevel)  
[JScript] public function SetAppDomainPolicy(domainPolicy : PolicyLevel);
```

Description

Establishes the security policy level for this application domain. The security policy level.

SetCachePath

[C#] public void SetCachePath(string path);

[C++] public: __sealed void SetCachePath(String* path);

[VB] NotOverridable Public Sub SetCachePath(ByVal path As String)

[JScript] public function SetCachePath(path : String);

Description

Establishes the specified directory path as the location where assemblies are shadow copied. The fully qualified path to the shadow copy location.

SetData

[C#] public void SetData(string name, object data);

[C++] public: __sealed void SetData(String* name, Object* data);

[VB] NotOverridable Public Sub SetData(ByVal name As String, ByVal data As Object)

[JScript] public function SetData(name : String, data : Object);

Description

Assigns the specified value to the specified application domain property.

This method has been superseded by properties of the

System.AppDomainSetup class. The following table shows the

System.AppDomainSetup property that corresponds to a value of *name* . The name of an application domain property. The value to set the *name* property.

SetDynamicBase

[C#] public void SetDynamicBase(string path);

[C++] public: void SetDynamicBase(String* path);

[VB] Public Sub SetDynamicBase(ByVal path As String)

[JScript] public function SetDynamicBase(path : String);

Description

Establishes the specified directory path as the location where dynamically generated files are stored and accessed.

This method sets the **System.AppDomainSetup.DynamicBase** property of the internal **System.AppDomainSetup** object associated with this instance. The fully qualified path to where dynamic assemblies are stored.

SetPrincipalPolicy

[C#] public void SetPrincipalPolicy(PrincipalPolicy policy);

[C++] public: __sealed void SetPrincipalPolicy(PrincipalPolicy policy);

[VB] NotOverridable Public Sub SetPrincipalPolicy(ByVal policy As PrincipalPolicy)

[JScript] public function SetPrincipalPolicy(policy : PrincipalPolicy);

Description

1 Set the class of the default principal object to be attached to threads if they
2 attempt to bind to a principal while executing in this application domain. The class
3 of the principal object to be attached to threads.

4 SetShadowCopyFiles

5
6 [C#] public void SetShadowCopyFiles();
7 [C++] public: void SetShadowCopyFiles();
8 [VB] Public Sub SetShadowCopyFiles()
9 [JScript] public function SetShadowCopyFiles();

10 *Description*

11 Turns on shadow copying.

12 SetShadowCopyPath

13
14
15 [C#] public void SetShadowCopyPath(string path);
16 [C++] public: __sealed void SetShadowCopyPath(String* path);
17 [VB] NotOverridable Public Sub SetShadowCopyPath(ByVal path As String)
18 [JScript] public function SetShadowCopyPath(path : String);

19 *Description*

20 Establishes the specified directory path as the location of assemblies to be
21 "shadow copied".

22 This method sets the **System.AppDomainSetup.ShadowCopyDirectories**
23 property of the internal **System.AppDomainSetup** object associated with this
24 instance. A list of directory names, where each name is separated by a semicolon.
25

SetThreadPrincipal

```
[C#] public void SetThreadPrincipal(IPrincipal principal);  
[C++] public: __sealed void SetThreadPrincipal(IPrincipal* principal);  
[VB] NotOverridable Public Sub SetThreadPrincipal(ByVal principal As  
IPrincipal)  
[JScript] public function SetThreadPrincipal(principal : IPrincipal);
```

Description

Set the default principal object to be attached to threads if they attempt to bind to a principal while executing in this application domain. The principal object to be attached to threads.

ToString

```
[C#] public override string ToString();  
[C++] public: String* ToString();  
[VB] Overrides Public Function ToString() As String  
[JScript] public override function ToString() : String;
```

Description

Obtains the **System.String** representation of the application domain.

Return Value: The friendly name, loader name, and loader context policy of the application domain.

The string representation specifies the friendly name of the application domain.

Unload

```
[C#] public static void Unload(AppDomain domain);
[C++] public: static void Unload(AppDomain* domain);
[VB] Public Shared Sub Unload(ByVal domain As AppDomain)
[JScript] public static function Unload(domain : AppDomain);
```

Description

Removes the specified application domain.

A long period of time can pass before *domain* unloads because it might be difficult to terminate executing threads. An application domain to be unloaded.

AppDomainSetup class (System)

Unload

Constructors:

AppDomainSetup

Example Syntax:

Unload

ApplicationBase

Unload

ApplicationName

Unload

CachePath

Unload

ConfigurationFile

Unload

- 1 DynamicBase
- 2 Unload
- 3 LicenseFile
- 4 Unload
- 5 LoaderOptimization
- 6 Unload
- 7 PrivateBinPath
- 8 Unload
- 9 PrivateBinPathProbe
- 10 Unload
- 11 ShadowCopyDirectories
- 12 Unload
- 13 ShadowCopyFiles
- 14 Unload
- 15 AppDomainUnloadedException class (System)
- 16 ToString

17
18

19 *Description*

20 The exception that is thrown when an attempt is made to access an
21 unloaded application domain.

22 **System.AppDomainUnloadedException** uses the HRESULT
23 COR_E_APPDOMAINUNLOADED, which has the value 0x80131014.

24 AppDomainUnloadedException

25 *Example Syntax:*

ToString

[C#] public AppDomainUnloadedException();

[C++] public: AppDomainUnloadedException();

[VB] Public Sub New()

[JScript] public function AppDomainUnloadedException(); Initializes a new instance of the **System.AppDomainUnloadedException** class.

Description

Initializes a new instance of the **System.AppDomainUnloadedException** class with default properties.

The following table shows the initial property values for an instance of **System.AppDomainUnloadedException**.

AppDomainUnloadedException

Example Syntax:

ToString

[C#] public AppDomainUnloadedException(string message);

[C++] public: AppDomainUnloadedException(String* message);

[VB] Public Sub New(ByVal message As String)

[JScript] public function AppDomainUnloadedException(message : String);

Description

Initializes a new instance of the **System.AppDomainUnloadedException** class with a specified error message.

The following table shows the initial property values for an instance of **System.AppDomainUnloadedException** . The error message that explains the reason for the exception.

AppDomainUnloadedException

Example Syntax:

ToString

[C#] protected AppDomainUnloadedException(SerializationInfo info, StreamingContext context);

[C++] protected: AppDomainUnloadedException(SerializationInfo* info, StreamingContext context);

[VB] Protected Sub New(ByVal info As SerializationInfo, ByVal context As StreamingContext)

[JScript] protected function AppDomainUnloadedException(info : SerializationInfo, context : StreamingContext);

Description

Initializes a new instance of the **System.AppDomainUnloadedException** class with serialized data.

This constructor is called during deserialization to reconstitute the exception object transmitted over a stream. For more information, see . The object that holds the serialized object data. The contextual information about the source or destination.

AppDomainUnloadedException

Example Syntax:

ToString

```
[C#] public AppDomainUnloadedException(string message, Exception
innerException);
[C++] public: AppDomainUnloadedException(String* message, Exception*
innerException);
[VB] Public Sub New(ByVal message As String, ByVal innerException As
Exception)
[JScript] public function AppDomainUnloadedException(message : String,
innerException : Exception);
```

Description

Initializes a new instance of the **System.AppDomainUnloadedException** class with a specified error message and a reference to the inner exception that is the root cause of this exception.

When an **Exception** *X* is thrown as a direct result of a previous exception *Y*, the **System.Exception.InnerException** property of *X* should contain a reference to *Y*. The **InnerException** property returns the same value as was passed into the constructor, or **null** if the inner exception value was not supplied to the constructor. The error message that explains the reason for the exception. An instance of **System.Exception** that is the cause of the current **Exception**. If *innerException* is non-null, then the current **Exception** is raised in a catch block handling *innerException*.

HelpLink

HResult

1 InnerException

2 Message

3 Source

4 StackTrace

5 TargetSite

6 ApplicationException class (System)

7 ToString

9
10 *Description*

11 The exception that is thrown when a non-fatal application error occurs.

12 **System.ApplicationException** is thrown by a user program, not by the
13 common language runtime. If you are designing an application that needs to create
14 its own exceptions, derive from the **System.ApplicationException** class.

15 ApplicationException

16 *Example Syntax:*

17 ToString

18
19 [C#] public ApplicationException();

20 [C++] public: ApplicationException();

21 [VB] Public Sub New()

22 [JScript] public function ApplicationException(); Initializes a new instance of the

23 **System.ApplicationException** class.

24
25 *Description*

1 Initializes a new instance of the **System.ApplicationException** class with
2 default properties.

3 The following table shows the initial property values for an instance of
4 **System.ApplicationException** .

5 ApplicationException

6 *Example Syntax:*

7 ToString

9 [C#] public ApplicationException(string message);

10 [C++] public: ApplicationException(String* message);

11 [VB] Public Sub New(ByVal message As String)

12 [JScript] public function ApplicationException(message : String);

14 *Description*

15 Initializes a new instance of the **System.ApplicationException** class with a
16 specified error message.

17 The following table shows the initial property values for an instance of
18 **System.ApplicationException** . The error message that explains the reason for
19 the exception.

20 ApplicationException

21 *Example Syntax:*

22 ToString

24 [C#] protected ApplicationException(SerializationInfo info, StreamingContext
25 context);

1 [C++] protected: ApplicationException(SerializationInfo* info, StreamingContext
2 context);

3 [VB] Protected Sub New(ByVal info As SerializationInfo, ByVal context As
4 StreamingContext)

5 [JScript] protected function ApplicationException(info : SerializationInfo, context
6 : StreamingContext);

7 8 *Description*

9 Initializes a new instance of the **System.ApplicationException** class with
10 serialized data.

11 This constructor is called during deserialization to reconstitute the
12 exception object transmitted over a stream. For more information, see . The object
13 that holds the serialized object data. The contextual information about the source
14 or destination.

15 ApplicationException

16 *Example Syntax:*

17 ToString

18
19 [C#] public ApplicationException(string message, Exception innerException);

20 [C++] public: ApplicationException(String* message, Exception*
21 innerException);

22 [VB] Public Sub New(ByVal message As String, ByVal innerException As
23 Exception)

24 [JScript] public function ApplicationException(message : String, innerException :
25 Exception);

Description

Initializes a new instance of the **System.ApplicationException** class with a specified error message and a reference to the inner exception that is the root cause of this exception.

When an **Exception** *X* is thrown as a direct result of a previous exception *Y*, the **System.Exception.InnerException** property of *X* should contain a reference to *Y*. The **InnerException** property returns the same value as was passed into the constructor, or **null** if the inner exception value was not supplied to the constructor. The error message that explains the reason for the exception. An instance of **System.Exception** that is the cause of the current **Exception**. If *innerException* is non-null, then the current **Exception** is raised in a catch block handling *innerException*.

HelpLink

HResult

InnerException

Message

Source

StackTrace

TargetSite

ArgIterator structure (System)

ToString

Description

Represents a variable-length argument list; that is, the parameters of a function that takes a variable number of arguments.

Typically, you use this class for writing compilers. The methods in this class are not generally useful in other kinds of applications.

ArgIterator

Example Syntax:

ToString

[C#] public ArgIterator(RuntimeArgumentHandle arglist);

[C++] public: ArgIterator(RuntimeArgumentHandle arglist);

[VB] Public Sub New(ByVal arglist As RuntimeArgumentHandle)

[JScript] public function ArgIterator(arglist : RuntimeArgumentHandle);

Initializes a new instance of the **ArgIterator** class.

Description

Initializes a new instance of the **ArgIterator** class using the specified argument list. An argument list consisting of both required and optional items.

ArgIterator

Example Syntax:

ToString

[C#] unsafe public ArgIterator(RuntimeArgumentHandle arglist, void* ptr);

[C++] public: ArgIterator(RuntimeArgumentHandle arglist, void* ptr);

End

1
2 [C#] public void End();

3 [C++] public: void End();

4 [VB] Public Sub End()

5 [JScript] public function End();

6
7 *Description*

8 Moves the iterator to the end of the variable-length argument list; that is, it
9 invalidates the iterator.

10 Conceptually, this method moves the iterator to the end of the list so that
11 the next call to **System.ArgIterator.GetNextArg** generates an exception.

12 **Equals**

13
14 [C#] public override bool Equals(object o);

15 [C++] public: bool Equals(Object* o);

16 [VB] Overrides Public Function Equals(ByVal o As Object) As Boolean

17 [JScript] public override function Equals(o : Object) : Boolean;

18
19 *Description*

20 This method is not supported, and always throws **NotSupportedException**
21 . An object to be compared to this instance.

22 **GetHashCode**

23
24 [C#] public override int GetHashCode();

25 [C++] public: int GetHashCode();

1 [VB] Overrides Public Function GetHashCode() As Integer

2 [JScript] public override function GetHashCode() : int;

3
4 *Description*

5 Returns the hash code of this object.

6 *Return Value:* A 32-bit signed integer hash code.

7 *GetNextArg*

8
9 [C#] public TypedReference GetNextArg();

10 [C++] public: TypedReference GetNextArg();

11 [VB] Public Function GetNextArg() As TypedReference

12 [JScript] public function GetNextArg() : TypedReference; Returns the next
13 argument in a variable-length argument list.

14
15 *Description*

16 Returns the next argument in a variable-length argument list.

17 *Return Value:* The next argument as a **System.TypedReference** object.

18 The iterator is automatically advanced to the next argument.

19 *GetNextArg*

20
21 [C#] public TypedReference GetNextArg(RuntimeTypeHandle rth);

22 [C++] public: TypedReference GetNextArg(RuntimeTypeHandle rth);

23 [VB] Public Function GetNextArg(ByVal rth As RuntimeTypeHandle) As

24 TypedReference

25 [JScript] public function GetNextArg(rth : RuntimeTypeHandle) :

1 TypedReference;

3 *Description*

4 Returns the next argument in a variable-length argument list that has a
5 specified type.

6 *Return Value:* The next argument as a **System.TypedReference** object.

7 The iterator is automatically advanced to the next argument. A runtime type
8 handle that identifies the type of the argument to retrieve.

9 **GetNextArgType**

11 [C#] public RuntimeTypeHandle GetNextArgType();

12 [C++] public: RuntimeTypeHandle GetNextArgType();

13 [VB] Public Function GetNextArgType() As RuntimeTypeHandle

14 [JScript] public function GetNextArgType() : RuntimeTypeHandle;

16 *Description*

17 Returns the type of the next argument.

18 *Return Value:* The type of the next argument.

19 This method does not advance the iterator to the next argument.

20 **GetRemainingCount**

22 [C#] public int GetRemainingCount();

23 [C++] public: int GetRemainingCount();

24 [VB] Public Function GetRemainingCount() As Integer

25 [JScript] public function GetRemainingCount() : int;

1
2 *Description*

3 Returns the number of arguments remaining in the argument list.

4 *Return Value:* The number of remaining arguments.

5 `ArgumentException` class (System)

6 `ToString`

7
8
9 *Description*

10 The exception that is thrown when one of the arguments provided to a
11 method is not valid.

12 **System.ArgumentException** is thrown when a method is invoked and at
13 least one of the passed arguments does not meet the parameter specification of the
14 called method. All instances of **System.ArgumentException** should carry a
15 meaningful error message describing the invalid argument, as well as the expected
16 range of values for the argument.

17 `ArgumentException`

18 *Example Syntax:*

19 `ToString`

20
21 `[C#] public ArgumentException();`

22 `[C++] public: ArgumentException();`

23 `[VB] Public Sub New()`

24 `[JScript] public function ArgumentException();` Initializes a new instance of the

25 **System.ArgumentException** class.

1
2 *Description*

3 Initializes a new instance of the **System.ArgumentException** class with
4 default properties.

5 The following table shows the initial property values for an instance of
6 **System.ArgumentException** .

7 ArgumentException

8 *Example Syntax:*

9 ToString

10
11 [C#] public ArgumentException(string message);

12 [C++] public: ArgumentException(String* message);

13 [VB] Public Sub New(ByVal message As String)

14 [JScript] public function ArgumentException(message : String);

15
16 *Description*

17 Initializes a new instance of the **System.ArgumentException** class with a
18 specified error message.

19 The following table shows the initial property values for an instance of
20 **System.ArgumentException** . The error message that explains the reason for the
21 exception.

22 ArgumentException

23 *Example Syntax:*

24 ToString

```

1
2 [C#] protected ArgumentException(SerializationInfo info, StreamingContext
3 context);
4 [C++] protected: ArgumentException(SerializationInfo* info, StreamingContext
5 context);
6 [VB] Protected Sub New(ByVal info As SerializationInfo, ByVal context As
7 StreamingContext)
8 [JScript] protected function ArgumentException(info : SerializationInfo, context :
9 StreamingContext);
10

```

Description

Initializes a new instance of the **System.ArgumentException** class with serialized data.

This constructor is called during deserialization to reconstitute the exception object transmitted over a stream. For more information, see . The object that holds the serialized object data. The contextual information about the source or destination.

ArgumentException

Example Syntax:

ToString

```

21
22 [C#] public ArgumentException(string message, Exception innerException);
23 [C++] public: ArgumentException(String* message, Exception* innerException);
24 [VB] Public Sub New(ByVal message As String, ByVal innerException As
25 Exception)

```

1 [JScript] public function ArgumentException(message : String, innerException :
2 Exception);

3
4 *Description*

5 Initializes a new instance of the **System.ArgumentException** class with a
6 specified error message and a reference to the inner exception that is the root cause
7 of this exception.

8 When an **Exception** *X* is thrown as a direct result of a previous exception *Y*,
9 the **System.Exception.InnerException** property of *X* should contain a reference
10 to *Y*. The **InnerException** property returns the same value as was passed into the
11 constructor, or **null** if the inner exception value was not supplied to the
12 constructor. The error message that explains the reason for the exception. An
13 instance of **System.Exception** that is the cause of the current **Exception**. If
14 *innerException* is non-null, then the current **Exception** is raised in a catch block
15 handling *innerException*.

16 ArgumentException

17 *Example Syntax:*

18 ToString

19
20 [C#] public ArgumentException(string message, string paramName);

21 [C++] public: ArgumentException(String* message, String* paramName);

22 [VB] Public Sub New(ByVal message As String, ByVal paramName As String)

23 [JScript] public function ArgumentException(message : String, paramName :
24 String);

Description

Initializes a new instance of the **System.ArgumentException** class with a specified error message and the name of the parameter that causes this exception.

The following table shows the initial property values for an instance of **System.ArgumentException**. The error message that explains the reason for the exception. The name of the invalid parameter.

ArgumentException

Example Syntax:

ToString

```
[C#] public ArgumentException(string message, string paramName, Exception  
innerException);
```

```
[C++] public: ArgumentException(String* message, String* paramName,  
Exception* innerException);
```

```
[VB] Public Sub New(ByVal message As String, ByVal paramName As String,  
ByVal innerException As Exception)
```

```
[JScript] public function ArgumentException(message : String, paramName :  
String, innerException : Exception);
```

Description

Initializes a new instance of the **System.ArgumentException** class with a specified error message, the parameter name, and a reference to the inner exception that is the root cause of this exception.

When an **Exception** *X* is thrown as a direct result of a previous exception *Y*, the **System.Exception.InnerException** property of *X* should contain a reference to *Y*. The **InnerException** property returns the same value as was passed into the constructor, or **null** if the inner exception value was not supplied to the constructor. The error message that explains the reason for the exception. The name of the invalid parameter. An instance of **System.Exception** that is the cause of the current **Exception**. If *innerException* is non-null, then the current **Exception** is raised in a catch block handling *innerException*.

HelpLink

HResult

InnerException

Message

ToString

Description

Gets the error message and the parameter name, or only the error message if no parameter name is set.

This property overrides **System.Exception.Message**. The error message should be localized.

ParamName

ToString

[C#] public virtual string ParamName {get;}

[C++] public: __property virtual String* get_ParamName();

1 [VB] Overridable Public ReadOnly Property ParamName As String

2 [JScript] public function get ParamName() : String;

3
4 *Description*

5 Gets the name of the parameter that causes this exception.

6 Every **System.ArgumentException** should carry the name of the
7 parameter that causes this exception. The parameter name should not be localized.

8 Source

9 StackTrace

10 TargetSite

11 GetObjectData

12
13 [C#] public override void GetObjectData(SerializationInfo info, StreamingContext
14 context);

15 [C++] public: void GetObjectData(SerializationInfo* info, StreamingContext
16 context);

17 [VB] Overrides Public Sub GetObjectData(ByVal info As SerializationInfo,
18 ByVal context As StreamingContext)

19 [JScript] public override function GetObjectData(info : SerializationInfo, context :
20 StreamingContext);

21
22 *Description*

23 Sets the **System.Runtime.Serialization.SerializationInfo** object with the
24 parameter name and additional exception information.

System.ArgumentException.GetObjectData(System.Runtime.Serialization.SerializationInfo, System.Runtime.Serialization.StreamingContext) sets a **System.Runtime.Serialization.SerializationInfo** with all the exception object data targeted for serialization. During deserialization, the exception object is reconstituted from the **System.Runtime.Serialization.SerializationInfo** transmitted over the stream. The object that holds the serialized object data. The contextual information about the source or destination.

ArgumentNullException class (System)

ToString

Description

The exception that is thrown when **null** is passed to a method that does not accept it as a valid argument.

System.ArgumentNullException uses the HRESULT E_POINTER, which has the value 0x80004003.

ArgumentNullException

Example Syntax:

ToString

[C#] public ArgumentNullException();

[C++] public: ArgumentNullException();

[VB] Public Sub New()

[JScript] public function ArgumentNullException(); Initializes a new instance of the **System.ArgumentNullException** class.

Description

Initializes a new instance of the **System.ArgumentNullException** class with default properties.

The following table shows the initial property values for an instance of **System.ArgumentNullException**.

ArgumentNullException

Example Syntax:

ToString

[C#] public ArgumentNullException(string paramName);

[C++] public: ArgumentNullException(String* paramName);

[VB] Public Sub New(ByVal paramName As String)

[JScript] public function ArgumentNullException(paramName : String);

Description

Initializes a new instance of the **System.ArgumentNullException** class with the name of the parameter that causes this exception.

The following table shows the initial property values for an instance of **System.ArgumentNullException**. The name of the parameter that is assigned **null**.

ArgumentNullException

Example Syntax:

ToString

1
2 [C#] protected ArgumentNullException(SerializationInfo info, StreamingContext
3 context);

4 [C++] protected: ArgumentNullException(SerializationInfo* info,
5 StreamingContext context);

6 [VB] Protected Sub New(ByVal info As SerializationInfo, ByVal context As
7 StreamingContext)

8 [JScript] protected function ArgumentNullException(info : SerializationInfo,
9 context : StreamingContext);

10 11 *Description*

12 Initializes a new instance of the **System.ArgumentNullException** class
13 with serialized data.

14 This constructor is called during deserialization to reconstitute the
15 exception object transmitted over a stream. For more information, see . The object
16 that holds the serialized object data. The contextual information about the source
17 or destination.

18 ArgumentNullException

19 *Example Syntax:*

20 ToString

21
22 [C#] public ArgumentNullException(string paramName, string message);

23 [C++] public: ArgumentNullException(String* paramName, String* message);

24 [VB] Public Sub New(ByVal paramName As String, ByVal message As String)

25 [JScript] public function ArgumentNullException(paramName : String, message :

String);

Description

Initializes an instance of the **System.ArgumentNullException** class with a specified error message and the name of the parameter that causes this exception.

The following table shows the initial property values for an instance of **System.ArgumentNullException**. The name of the parameter that is assigned **null**. The error message that explains the reason for the exception.

HelpLink

HResult

InnerException

Message

ParamName

Source

StackTrace

TargetSite

ArgumentOutOfRangeException class (System)

ToString

Description

The exception that is thrown when the value of an argument is outside the allowable range of values as defined by the invoked method.

System.ArgumentOutOfRangeException is used extensively by: Classes in the **System.Collections** and **System.IO** namespaces.

ArgumentOutOfRangeException

Example Syntax:

ToString

[C#] public ArgumentOutOfRangeException();

[C++] public: ArgumentOutOfRangeException();

[VB] Public Sub New()

[JScript] public function ArgumentOutOfRangeException(); Initializes a new instance of the **System.ArgumentOutOfRangeException** class.

Description

Initializes a new instance of the **System.ArgumentOutOfRangeException** class with default properties.

The following table shows the initial property values for an instance of **System.ArgumentOutOfRangeException**.

ArgumentOutOfRangeException

Example Syntax:

ToString

[C#] public ArgumentOutOfRangeException(string paramName);

[C++] public: ArgumentOutOfRangeException(String* paramName);

[VB] Public Sub New(ByVal paramName As String)

[JScript] public function ArgumentOutOfRangeException(paramName : String);

Description

1 Initializes a new instance of the **System.ArgumentOutOfRangeException**
2 class with the name of the parameter that causes this exception.

3 The following table shows the initial property values for an instance of
4 **System.ArgumentOutOfRangeException** . The name of the invalid parameter.

5 ArgumentOutOfRangeException

6 *Example Syntax:*

7 ToString

9 [C#] protected ArgumentOutOfRangeException(SerializationInfo info,
10 StreamingContext context);

11 [C++] protected: ArgumentOutOfRangeException(SerializationInfo* info,
12 StreamingContext context);

13 [VB] Protected Sub New(ByVal info As SerializationInfo, ByVal context As
14 StreamingContext)

15 [JScript] protected function ArgumentOutOfRangeException(info :
16 SerializationInfo, context : StreamingContext);

18 *Description*

19 Initializes a new instance of the **System.ArgumentOutOfRangeException**
20 class with serialized data.

21 This constructor is called during deserialization to reconstitute the
22 exception object transmitted over a stream. For more information, see . The object
23 that holds the serialized object data. The contextual information about the source
24 or destination.

25 ArgumentOutOfRangeException

Example Syntax:

ToString

```
[C#] public ArgumentOutOfRangeException(string paramName, string message);  
[C++] public: ArgumentOutOfRangeException(String* paramName, String*  
message);  
[VB] Public Sub New(ByVal paramName As String, ByVal message As String)  
[JScript] public function ArgumentOutOfRangeException(paramName : String,  
message : String);
```

Description

Initializes a new instance of the **System.ArgumentOutOfRangeException** class with a specified error message and the name of the parameter that causes this exception.

The following table shows the initial property values for an instance of **System.ArgumentOutOfRangeException** . The name of the invalid parameter.
The error message that explains the reason for the exception.

ArgumentOutOfRangeException

Example Syntax:

ToString

```
[C#] public ArgumentOutOfRangeException(string paramName, object  
actualValue, string message);  
[C++] public: ArgumentOutOfRangeException(String* paramName, Object*  
actualValue, String* message);
```



```

1 [VB] Public Sub New(ByVal paramName As String, ByVal actualValue As
2 Object, ByVal message As String)
3 [JScript] public function ArgumentOutOfRangeException(paramName : String,
4 actualValue : Object, message : String);

```

Description

Initializes a new instance of the **System.ArgumentOutOfRangeException** class with a specified error message, the parameter name, and the value of the argument.

This constructor with the additional argument *actualValue* is not used within the .NET Framework class library. The **System.ArgumentOutOfRangeException.ActualValue** property is provided so that applications can make use of the available argument value. The name of the invalid parameter. The value of the argument that causes this exception. The error message that explains the reason for the exception.

ActualValue

ToString

```

19 [C#] public virtual object ActualValue {get;}
20 [C++] public: __property virtual Object* get_ActualValue();
21 [VB] Overridable Public ReadOnly Property ActualValue As Object
22 [JScript] public function get ActualValue() : Object;

```

Description

Gets the argument value that causes this exception.

The **System.ArgumentOutOfRangeException.ActualValue** property is assigned a value at the time of object construction. If the **System.ArgumentOutOfRangeException.ActualValue** property value is not **null**, a string representation of the value is then appended to the message string held by the **System.ArgumentOutOfRangeException.Message** property.

HelpLink

HResult

InnerException

Message

ToString

Description

Gets the error message and the string representation of the invalid argument value, or only the error message if the argument value is null.

This property overrides **System.ArgumentException.Message**.

ParamName

Source

StackTrace

TargetSite

GetObjectData

```
[C#] public override void GetObjectData(SerializationInfo info, StreamingContext context);
```

```
[C++] public: void GetObjectData(SerializationInfo* info, StreamingContext
```

1 context);
2 [VB] Overrides Public Sub GetObjectData(ByVal info As SerializationInfo,
3 ByVal context As StreamingContext)
4 [JScript] public override function GetObjectData(info : SerializationInfo, context :
5 StreamingContext);
6

7 *Description*

8 Sets the **System.Runtime.Serialization.SerializationInfo** object with the
9 invalid argument value and additional exception information.

10 **System.ArgumentOutOfRangeException.GetObjectData(System.Runti**
11 **me.Serialization.SerializationInfo,System.Runtime.Serialization.StreamingCo**
12 **ntext)** sets a **System.Runtime.Serialization.SerializationInfo** with all the
13 exception object data targeted for serialization. During deserialization, the
14 exception object is reconstituted from the
15 **System.Runtime.Serialization.SerializationInfo** transmitted over the stream.
16 The object that holds the serialized object data. The contextual information about
17 the source or destination.

18 ArithmeticException class (System)

19 ToString
20
21

22 *Description*

23 The exception that is thrown for errors in an arithmetic, casting, or
24 conversion operation.
25

System.ArithmeticException is the base class for **System.DivideByZeroException** , **System.NotFiniteNumberException** , and **System.OverflowException** . In general, use one of the derived classes of **System.ArithmeticException** to more precisely indicate the exact nature of the error. Throw an **System.ArithmeticException** if you are only interested in capturing a general arithmetic error.

ArithmeticException

Example Syntax:

ToString

[C#] public ArithmeticException();

[C++] public: ArithmeticException();

[VB] Public Sub New()

[JScript] public function ArithmeticException(); Initializes a new instance of the **System.ArithmeticException** class.

Description

Initializes a new instance of the **System.ArithmeticException** class with default properties.

The following table shows initial property values for an instance of **System.ArithmeticException** .

ArithmeticException

Example Syntax:

ToString

```

1
2 [C#] public ArithmeticException(string message);
3 [C++] public: ArithmeticException(String* message);
4 [VB] Public Sub New(ByVal message As String)
5 [JScript] public function ArithmeticException(message : String);
6

```

Description

Initializes a new instance of the **System.ArithmeticException** class with a specified error message.

The following table shows initial property values for an instance of **System.ArithmeticException** . The error message that explains the reason for the exception.

ArithmeticException

Example Syntax:

ToString

```

17 [C#] protected ArithmeticException(SerializationInfo info, StreamingContext
18 context);
19 [C++] protected: ArithmeticException(SerializationInfo* info, StreamingContext
20 context);
21 [VB] Protected Sub New(ByVal info As SerializationInfo, ByVal context As
22 StreamingContext)
23 [JScript] protected function ArithmeticException(info : SerializationInfo, context :
24 StreamingContext);
25

```

Description

Initializes a new instance of the **System.ArithmeticException** class with serialized data.

This constructor is called during deserialization to reconstitute the exception object transmitted over a stream. For more information, see . The object that holds the serialized object data. The contextual information about the source or destination.

ArithmeticException

Example Syntax:

ToString

```
[C#] public ArithmeticException(string message, Exception innerException);
```

```
[C++] public: ArithmeticException(String* message, Exception* innerException);
```

```
[VB] Public Sub New(ByVal message As String, ByVal innerException As  
Exception)
```

```
[JScript] public function ArithmeticException(message : String, innerException :  
Exception);
```

Description

Initializes a new instance of the **System.ArithmeticException** class with a specified error message and a reference to the exception that is the root cause of this exception.

When an **Exception** *X* is thrown as a direct result of a previous exception *Y*, the **System.Exception.InnerException** property of *X* should contain a reference

1 to *Y* . The **InnerException** property returns the same value as was passed into the
2 constructor, or **null** if the inner exception value was not supplied to the
3 constructor. The error message that explains the reason for the exception. An
4 instance of **System.Exception** that is the cause of the current **Exception**. If
5 *innerException* is non-null, then the current **Exception** is raised in a catch block
6 handling *innerException* .

7 HelpLink

8 HResult

9 InnerException

10 Message

11 Source

12 StackTrace

13 TargetSite

14 Array class (System)

15 ToString

16
17
18 *Description*

19 Provides methods for creating, manipulating, searching and sorting arrays,
20 thereby serving as the base class for all arrays in the common language runtime.

21 An element is a value in the **System.Array** . The length of an
22 **System.Array** is the total number of elements it can contain. The rank of an
23 **System.Array** is the number of dimensions in the **System.Array** . The lower
24 bound or lowbound of a dimension of an **System.Array** is the starting index of
25

that dimension of the **System.Array** ; a multidimensional **System.Array** can have different bounds for each dimension.

Array

Example Syntax:

ToString

[C#] protected Array();

[C++] protected: Array();

[VB] Protected Sub New()

[JScript] protected function Array();

IsFixedSize

ToString

[C#] public virtual bool IsFixedSize {get;}

[C++] public: __property virtual bool get_IsFixedSize();

[VB] Overridable Public ReadOnly Property IsFixedSize As Boolean

[JScript] public function get IsFixedSize() : Boolean;

Description

Gets a value indicating whether the **System.Array** has a fixed size.

This method implements the **System.Collections.ICollection** interface. It can be overridden by a derived class.

IsReadOnly

ToString


```

1
2 [C#] public virtual bool IsReadOnly {get;}
3 [C++] public: __property virtual bool get_IsReadOnly();
4 [VB] Overridable Public ReadOnly Property IsReadOnly As Boolean
5 [JScript] public function get IsReadOnly() : Boolean;
6

```

Description

Gets a value indicating whether the **System.Array** is read-only.

This method implements the **System.Collections.ICollection** interface. It can be overridden by a derived class.

IsSynchronized

ToString

```

13
14 [C#] public virtual bool IsSynchronized {get;}
15 [C++] public: __property virtual bool get_IsSynchronized();
16 [VB] Overridable Public ReadOnly Property IsSynchronized As Boolean
17 [JScript] public function get IsSynchronized() : Boolean;
18

```

Description

Gets a value indicating whether access to the **System.Array** is synchronized (thread-safe).

This property implements the **System.Collections.ICollection** interface.

Length

ToString

```

1
2 [C#] public int Length {get;}
3 [C++] public: __property int get_Length();
4 [VB] Public ReadOnly Property Length As Integer
5 [JScript] public function get Length() : int;
6

```

Description

Gets the total number of elements in all the dimensions of the

System.Array .

Rank

ToString

```

13 [C#] public int Rank {get;}
14 [C++] public: __property int get_Rank();
15 [VB] Public ReadOnly Property Rank As Integer
16 [JScript] public function get Rank() : int;
17

```

Description

Gets the rank (number of dimensions) of the **System.Array** .

SyncRoot

ToString

```

23 [C#] public virtual object SyncRoot {get;}
24 [C++] public: __property virtual Object* get_SyncRoot();
25 [VB] Overridable Public ReadOnly Property SyncRoot As Object

```

1 [JScript] public function get SyncRoot() : Object;

3 *Description*

4 Gets an object that can be used to synchronize access to the **System.Array**

6 This property implements the **System.Collections.ICollection** interface.

7 BinarySearch

9 [C#] public static int BinarySearch(Array array, object value);

10 [C++] public: static int BinarySearch(Array* array, Object* value);

11 [VB] Public Shared Function BinarySearch(ByVal array As Array, ByVal value
12 As Object) As Integer

13 [JScript] public static function BinarySearch(array : Array, value : Object) : int;

14 Searches a one-dimensional sorted **System.Array** for a value, using a binary
15 search algorithm.

17 *Description*

18 Searches a one-dimensional sorted **System.Array** for a specific element,
19 using the **System.IComparable** interface implemented by each element of the
20 **System.Array** and by the specified **System.Object** .

21 *Return Value:* The index of *value* in the **System.Array** , if *value* is found;
22 otherwise, a negative number, which is the bitwise complement of the index of the
23 first element that is larger than *value* .

24 The *value* parameter and each element of *array* must implement the
25 **System.IComparable** interface, which is used for comparisons. If *array* 's

elements are not already sorted in increasing value according to the **System.IComparable** implementation, the result might be incorrect. Duplicate elements are allowed. The one-dimensional **System.Array** to search. The **System.Object** to search for.

BinarySearch

[C#] public static int BinarySearch(Array array, object value, IComparer comparer);

[C++] public: static int BinarySearch(Array* array, Object* value, IComparer* comparer);

[VB] Public Shared Function BinarySearch(ByVal array As Array, ByVal value As Object, ByVal comparer As IComparer) As Integer

[JScript] public static function BinarySearch(array : Array, value : Object, comparer : IComparer) : int;

Description

Searches a one-dimensional sorted **System.Array** for a value, using the specified **System.Collections.IComparer** interface.

Return Value: The index of *value* in the **System.Array** , if *value* is found; otherwise, a negative number, which is the bitwise complement of the index of the first element that is larger than *value* .

The comparer customizes how the elements are compared. For example, you can use a **System.Collections.CaseInsensitiveComparer** instance as the comparer to perform case-insensitive string searches. The one-dimensional **System.Array** to search. The **System.Object** to search for. The

System.Collections.IComparer implementation to use when comparing elements.

BinarySearch

[C#] public static int BinarySearch(Array array, int index, int length, object value);

[C++] public: static int BinarySearch(Array* array, int index, int length, Object* value);

[VB] Public Shared Function BinarySearch(ByVal array As Array, ByVal index As Integer, ByVal length As Integer, ByVal value As Object) As Integer

[JScript] public static function BinarySearch(array : Array, index : int, length : int, value : Object) : int;

Description

Searches a section of a one-dimensional sorted **System.Array** for a value, using the **System.IComparable** interface implemented by each element of the **System.Array** and by the specified value.

Return Value: The index of *value* in the **System.Array** , if *value* is found; otherwise, a negative number, which is the bitwise complement of the index of the first element that is larger than *value* .

The *value* parameter and each element of *array* must implement the **System.IComparable** interface, which is used for comparisons. If *array* 's elements are not already sorted in increasing value according to the **System.IComparable** implementation, the result might be incorrect. Duplicate elements are allowed. The one-dimensional **System.Array** to search. The starting

index of the range to search. The length of the range to search. The

System.Object to search for.

BinarySearch

[C#] public static int BinarySearch(Array array, int index, int length, object value, IComparer comparer);

[C++] public: static int BinarySearch(Array* array, int index, int length, Object* value, IComparer* comparer);

[VB] Public Shared Function BinarySearch(ByVal array As Array, ByVal index As Integer, ByVal length As Integer, ByVal value As Object, ByVal comparer As IComparer) As Integer

[JScript] public static function BinarySearch(array : Array, index : int, length : int, value : Object, comparer : IComparer) : int;

Description

Searches a section of a one-dimensional sorted **System.Array** for a value, using the specified **System.Collections.IComparer** interface.

Return Value: The index of *value* in the **System.Array** , if *value* is found; otherwise, a negative number, which is the bitwise complement of the index of the first element that is larger than *value* .

The comparer customizes how the elements are compared. For example, you can use a **System.Collections.CaseInsensitiveComparer** instance as the comparer to perform case-insensitive string searches. The one-dimensional **System.Array** to search. The starting index of the range to search. The length of the range to search. The **System.Object** to search for. The

System.Collections.IComparer implementation to use when comparing elements.

Clear

[C#] public static void Clear(Array array, int index, int length);

[C++] public: static void Clear(Array* array, int index, int length);

[VB] Public Shared Sub Clear(ByVal array As Array, ByVal index As Integer, ByVal length As Integer)

[JScript] public static function Clear(array : Array, index : int, length : int);

Description

Sets a range of elements in the **System.Array** to zero or to **null** .

Reference-type elements are set to **null** . Value-type elements are set to zero. The **System.Array** whose elements need to be cleared. The starting index of the range of elements to clear. The number of elements to clear.

Clone

[C#] public virtual object Clone();

[C++] public: virtual Object* Clone();

[VB] Overridable Public Function Clone() As Object

[JScript] public function Clone() : Object;

Description

Creates a shallow copy of the **System.Array** .

Return Value: A shallow copy of the **System.Array** .

This method can be overridden by a derived class.

Copy

```
[C#] public static void Copy(Array sourceArray, Array destinationArray, int  
length);
```

```
[C++] public: static void Copy(Array* sourceArray, Array* destinationArray, int  
length);
```

```
[VB] Public Shared Sub Copy(ByVal sourceArray As Array, ByVal  
destinationArray As Array, ByVal length As Integer)
```

```
[JScript] public static function Copy(sourceArray : Array, destinationArray :  
Array, length : int); Copies a section of one System.Array to another  
System.Array and performs type downcasting as required.
```

Description

Copies a range of elements from an **System.Array** starting at the first element and pastes them into another **System.Array** starting at the first element.

sourceArray and *destinationArray* must have the same number of dimensions. The **System.Array** that contains the data to copy. The **System.Array** that receives the data. The number of elements to copy.

Copy

```
[C#] public static void Copy(Array sourceArray, int sourceIndex, Array  
destinationArray, int destinationIndex, int length);
```

```
[C++] public: static void Copy(Array* sourceArray, int sourceIndex, Array*  
destinationArray, int destinationIndex, int length);
```


[VB] Public Shared Sub Copy(ByVal sourceArray As Array, ByVal sourceIndex As Integer, ByVal destinationArray As Array, ByVal destinationIndex As Integer, ByVal length As Integer)

[JScript] public static function Copy(sourceArray : Array, sourceIndex : int, destinationArray : Array, destinationIndex : int, length : int);

Description

Copies a range of elements from an **System.Array** starting at the specified source index and pastes them to another **System.Array** starting at the specified destination index.

sourceArray and *destinationArray* must have the same number of dimensions. The **System.Array** that contains the data to copy. The index in the *sourceArray* at which copying begins. The **System.Array** that receives the data. The index in the *destinationArray* at which storing begins. The number of elements to copy.

CopyTo

[C#] public virtual void CopyTo(Array array, int index);

[C++] public: virtual void CopyTo(Array* array, int index);

[VB] Overridable Public Sub CopyTo(ByVal array As Array, ByVal index As Integer)

[JScript] public function CopyTo(array : Array, index : int);

Description

Copies all the elements of the current one-dimensional **System.Array** to the specified one-dimensional **System.Array** starting at the specified destination **System.Array** index.

This method can be overridden by a derived class. The one-dimensional **System.Array** that is the destination of the elements copied from the current **System.Array**. The zero-based relative index in *array* at which copying begins.

CreateInstance

[C#] public static Array CreateInstance(Type elementType, int length);
[C++] public: static Array* CreateInstance(Type* elementType, int length);
[VB] Public Shared Function CreateInstance(ByVal elementType As Type, ByVal length As Integer) As Array
[JScript] public static function CreateInstance(elementType : Type, length : int) : Array; Initializes a new instance of the **System.Array** class.

Description

Creates a one-dimensional **System.Array** of the specified **System.Type** and length, with zero-based indexing.

Return Value: A new one-dimensional **System.Array** of the specified **System.Type** with the specified length, using zero-based indexing.

Unlike most classes, **System.Array** provides the **System.Array.CreateInstance(System.Type, System.Int32)** method, instead of public constructors, to allow for late bound access. The **System.Type** of **System.Array** to create. The size of the **System.Array** to create.

CreateInstance

```

1
2 [C#] public static Array CreateInstance(Type elementType, int[] lengths);
3 [C++] public: static Array* CreateInstance(Type* elementType, int lengths
4 __gc[]);
5 [VB] Public Shared Function CreateInstance(ByVal elementType As Type, ByVal
6 lengths() As Integer) As Array
7 [JScript] public static function CreateInstance(elementType : Type, lengths : int[])
8 : Array;
9

```

Description

Creates a multidimensional **System.Array** of the specified **System.Type** and dimension lengths, with zero-based indexing.

Return Value: A new multidimensional **System.Array** of the specified **System.Type** with the specified length for each dimension, using zero-based indexing.

Unlike most classes, **System.Array** provides the **System.Array.CreateInstance(System.Type, System.Int32)** method, instead of public constructors, to allow for late bound access. The **System.Type** of **System.Array** to create. An array that contains the size of each dimension of the **System.Array** to create.

CreateInstance

```

23 [C#] public static Array CreateInstance(Type elementType, int length1, int
24 length2);
25 [C++] public: static Array* CreateInstance(Type* elementType, int length1, int

```

```

1 length2);
2 [VB] Public Shared Function CreateInstance(ByVal elementType As Type, ByVal
3 length1 As Integer, ByVal length2 As Integer) As Array
4 [JScript] public static function CreateInstance(elementType : Type, length1 : int,
5 length2 : int) : Array;

```

Description

Creates a two-dimensional **System.Array** of the specified **System.Type** and dimension lengths, with zero-based indexing.

Return Value: A new two-dimensional **System.Array** of the specified **System.Type** with the specified length for each dimension, using zero-based indexing.

Unlike most classes, **System.Array** provides the **System.Array.CreateInstance(System.Type, System.Int32)** method, instead of public constructors, to allow for late bound access. The **System.Type** of **System.Array** to create. The size of the first dimension of the **System.Array** to create. The size of the second dimension of the **System.Array** to create.

CreateInstance

```

20 [C#] public static Array CreateInstance(Type elementType, int[] lengths, int[]
21 lowerBounds);
22 [C++] public: static Array* CreateInstance(Type* elementType, int lengths
23 __gc[], int lowerBounds __gc[]);
24 [VB] Public Shared Function CreateInstance(ByVal elementType As Type, ByVal
25 lengths() As Integer, ByVal lowerBounds() As Integer) As Array

```

1 [JScript] public static function CreateInstance(elementType : Type, lengths : int[],
2 lowerBounds : int[]) : Array;

4 *Description*

5 Creates a multidimensional **System.Array** of the specified **System.Type**
6 and dimension lengths, with the specified lower bounds.

7 *Return Value:* A new multidimensional **System.Array** of the specified
8 **System.Type** with the specified length and lower bound for each dimension.

9 Unlike most classes, **System.Array** provides the
10 **System.Array.CreateInstance(System.Type, System.Int32)** method, instead of
11 public constructors, to allow for late bound access. The **System.Type** of
12 **System.Array** to create. A one-dimensional array that contains the size of each
13 dimension of the **System.Array** to create. A one-dimensional array that contains
14 the lower bound (starting index) of each dimension of the **System.Array** to create.

15 *CreateInstance*

16
17 [C#] public static Array CreateInstance(Type elementType, int length1, int
18 length2, int length3);

19 [C++] public: static Array* CreateInstance(Type* elementType, int length1, int
20 length2, int length3);

21 [VB] Public Shared Function CreateInstance(ByVal elementType As Type, ByVal
22 length1 As Integer, ByVal length2 As Integer, ByVal length3 As Integer) As
23 Array

24 [JScript] public static function CreateInstance(elementType : Type, length1 : int,
25 length2 : int, length3 : int) : Array;

Description

Creates a three-dimensional **System.Array** of the specified **System.Type** and dimension lengths, with zero-based indexing.

Return Value: A new three-dimensional **System.Array** of the specified **System.Type** with the specified length for each dimension, using zero-based indexing.

Unlike most classes, **System.Array** provides the **System.Array.CreateInstance(System.Type, System.Int32)** method, instead of public constructors, to allow for late bound access. The **System.Type** of **System.Array** to create. The size of the first dimension of the **System.Array** to create. The size of the second dimension of the **System.Array** to create. The size of the third dimension of the **System.Array** to create.

GetEnumerator

[C#] public virtual IEnumerator GetEnumerator();

[C++] public: virtual IEnumerator* GetEnumerator();

[VB] Overridable Public Function GetEnumerator() As IEnumerator

[JScript] public function GetEnumerator() : IEnumerator;

Description

Returns an **System.Collections.IEnumerator** for the **System.Array**.

Return Value: An **System.Collections.IEnumerator** for the **System.Array**.

This method can be overridden by a derived class.

GetLength

1
2 [C#] public int GetLength(int dimension);

3 [C++] public: int GetLength(int dimension);

4 [VB] Public Function GetLength(ByVal dimension As Integer) As Integer

5 [JScript] public function GetLength(dimension : int) : int;

6
7 *Description*

8 Gets the number of elements in the specified dimension of the

9 **System.Array** .

10 *Return Value:* The number of elements in the specified dimension.

11 For example, GetLength(0) returns the number of elements in the first
12 dimension of the **System.Array** . A zero-based dimension of the **System.Array**
13 whose length needs to be determined.

14 GetLowerBound

15
16 [C#] public int GetLowerBound(int dimension);

17 [C++] public: int GetLowerBound(int dimension);

18 [VB] Public Function GetLowerBound(ByVal dimension As Integer) As Integer

19 [JScript] public function GetLowerBound(dimension : int) : int;

20
21 *Description*

22 Gets the lower bound of the specified dimension in the **System.Array** .

23 *Return Value:* The lower bound of the specified dimension in the **System.Array** .

24 For example, GetLowerBound(0) returns the lower bound for the indexes
25 of the first dimension of the **System.Array** , and GetLowerBound(Rank - 1)

returns the lower bound of the last dimension of the **System.Array** . A zero-based dimension of the **System.Array** whose lower bound needs to be determined.

GetUpperBound

[C#] public int GetUpperBound(int dimension);

[C++] public: int GetUpperBound(int dimension);

[VB] Public Function GetUpperBound(ByVal dimension As Integer) As Integer

[JScript] public function GetUpperBound(dimension : int) : int;

Description

Gets the upper bound of the specified dimension in the **System.Array** .

Return Value: The upper bound of the specified dimension in the **System.Array** .

For example, GetUpperBound(0) returns the upper bound for the indexes of the first dimension of the **System.Array** and GetUpperBound(Rank - 1) returns the upper bound of the last dimension of the **System.Array** . A zero-based dimension of the **System.Array** whose upper bound needs to be determined.

GetValue

[C#] public object GetValue(int index);

[C++] public: Object* GetValue(int index);

[VB] Public Function GetValue(ByVal index As Integer) As Object

[JScript] public function GetValue(index : int) : Object;

Description

Gets the value at the specified position in a one-dimensional **System.Array**

Return Value: The value at the specified position in the one-dimensional

System.Array .

The **System.Array.GetLowerBound(System.Int32)** and **System.Array.GetUpperBound(System.Int32)** methods can determine whether the value of *index* is out of bounds. The position of the value to get from the **System.Array**.

GetValue

[C#] public object GetValue(int[] indices);

[C++] public: Object* GetValue(int indices __gc[]);

[VB] Public Function GetValue(ByVal indices() As Integer) As Object

[JScript] public function GetValue(indices : int[]) : Object; Gets the values of the **System.Array** elements at the specified indexes.

Description

Gets the value at the specified position in a multidimensional **System.Array** .

Return Value: The value at the specified position in the **System.Array** .

The number of elements in *indices* must equal the number of dimensions in the **System.Array** . All elements in the *indices* array must collectively specify the position of the desired element in the multidimensional **System.Array** . A one-dimensional array of indexes that specifies the position of the element to get from the **System.Array**.

GetValue

```
[C#] public object GetValue(int index1, int index2);  
[C++] public: Object* GetValue(int index1, int index2);  
[VB] Public Function GetValue(ByVal index1 As Integer, ByVal index2 As  
Integer) As Object  
[JScript] public function GetValue(index1 : int, index2 : int) : Object;
```

Description

Gets the value at the specified position in a two-dimensional **System.Array**

.

Return Value: The value at the specified position in the **System.Array** .

The **System.Array.GetLowerBound(System.Int32)** and **System.Array.GetUpperBound(System.Int32)** methods can determine whether any of the indexes is out of bounds. The first-dimension index of the **System.Array** element to get. The second-dimension index of the **System.Array** element to get.

GetValue

```
[C#] public object GetValue(int index1, int index2, int index3);  
[C++] public: Object* GetValue(int index1, int index2, int index3);  
[VB] Public Function GetValue(ByVal index1 As Integer, ByVal index2 As  
Integer, ByVal index3 As Integer) As Object  
[JScript] public function GetValue(index1 : int, index2 : int, index3 : int) : Object;
```

1
2 *Description*

3 Gets the value at the specified position in a three-dimensional

4 **System.Array** .

5 *Return Value:* The value at the specified position in the **System.Array** .

6 The **System.Array.GetLowerBound(System.Int32)** and
7 **System.Array.GetUpperBound(System.Int32)** methods can determine whether
8 any of the indexes is out of bounds. The first-dimension index of the
9 **System.Array** element to get. The second-dimension index of the **System.Array**
10 element to get. The third-dimension index of the **System.Array** element to get.

11 **IndexOf**

12
13 [C#] public static int IndexOf(Array array, object value);

14 [C++] public: static int IndexOf(Array* array, Object* value);

15 [VB] Public Shared Function IndexOf(ByVal array As Array, ByVal value As
16 Object) As Integer

17 [JScript] public static function IndexOf(array : Array, value : Object) : int; Returns
18 the index of the first occurrence of a value in a one-dimensional **System.Array** or
19 in a portion of the **System.Array** .

20
21 *Description*

22 Searches for the specified **System.Object** and returns the index of the first
23 occurrence within the entire one-dimensional **System.Array** .

24 *Return Value:* The index of the first occurrence of *value* within the entire *array* , if
25 found; otherwise, the lower bound of the array - 1.

The one-dimensional **System.Array** is searched forward starting at the first element and ending at the last element. The elements are compared to the specified value using the **System.Object.Equals(System.Object)** method. The one-dimensional **System.Array** to search. The **System.Object** to locate in *array*.

IndexOf

```
[C#] public static int IndexOf(Array array, object value, int startIndex);  
[C++] public: static int IndexOf(Array* array, Object* value, int startIndex);  
[VB] Public Shared Function IndexOf(ByVal array As Array, ByVal value As  
Object, ByVal startIndex As Integer) As Integer  
[JScript] public static function IndexOf(array : Array, value : Object, startIndex :  
int) : int;
```

Description

Searches for the specified **System.Object** and returns the index of the first occurrence within the section of the one-dimensional **System.Array** that extends from the specified index to the last element.

Return Value: The index of the first occurrence of *value* within the section of *array* that extends from *startIndex* to the last element, if found; otherwise, the lower bound of the array - 1.

The one-dimensional **System.Array** is searched forward starting at *startIndex* and ending at the last element. The elements are compared to the specified value using the **System.Object.Equals(System.Object)** method. The one-dimensional **System.Array** to search. The **System.Object** to locate in *array*.
The starting index of the search.

IndexOf

```
[C#] public static int IndexOf(Array array, object value, int startIndex, int count);  
[C++] public: static int IndexOf(Array* array, Object* value, int startIndex, int  
count);  
[VB] Public Shared Function IndexOf(ByVal array As Array, ByVal value As  
Object, ByVal startIndex As Integer, ByVal count As Integer) As Integer  
[JScript] public static function IndexOf(array : Array, value : Object, startIndex :  
int, count : int) : int;
```

Description

Searches for the specified **System.Object** and returns the index of the first occurrence within the section of the one-dimensional **System.Array** that starts at the specified index and contains the specified number of elements.

Return Value: The index of the first occurrence of *value* within the section of *array* that starts at *startIndex* and contains *count* number of elements, if found; otherwise, the lower bound of the array - 1.

The one-dimensional **System.Array** is searched forward starting at *startIndex* and ending at *startIndex* + *count* - 1. The elements are compared to the specified value using the **System.Object.Equals(System.Object)** method. The one-dimensional **System.Array** to search. The **System.Object** to locate in *array*. The starting index of the search. The number of elements in the section to search.

Initialize

```
[C#] public void Initialize();
```

1 [C++] public: void Initialize();
2 [VB] Public Sub Initialize()
3 [JScript] public function Initialize();

4
5 *Description*

6 Initializes every element of the value-type **System.Array** by calling the
7 default constructor of the value type.

8 This method must not be used on reference-type arrays.

9 LastIndexOf

10
11 [C#] public static int LastIndexOf(Array array, object value);
12 [C++] public: static int LastIndexOf(Array* array, Object* value);
13 [VB] Public Shared Function LastIndexOf(ByVal array As Array, ByVal value As
14 Object) As Integer
15 [JScript] public static function LastIndexOf(array : Array, value : Object) : int;

16 Returns the index of the last occurrence of a value in a one-dimensional
17 **System.Array** or in a portion of the **System.Array** .

18
19 *Description*

20 Searches for the specified **System.Object** and returns the index of the last
21 occurrence within the entire one-dimensional **System.Array** .

22 *Return Value:* The index of the last occurrence of *value* within the entire *array* , if
23 found; otherwise, the lower bound of the array - 1.

24 The one-dimensional **System.Array** is searched backward starting at the
25 last element and ending at the first element. The elements are compared to the

specified value using the **System.Object.Equals(System.Object)** method. The one-dimensional **System.Array** to search. The **System.Object** to locate in *array*.

LastIndexOf

[C#] public static int LastIndexOf(Array array, object value, int startIndex);

[C++] public: static int LastIndexOf(Array* array, Object* value, int startIndex);

[VB] Public Shared Function LastIndexOf(ByVal array As Array, ByVal value As Object, ByVal startIndex As Integer) As Integer

[JScript] public static function LastIndexOf(array : Array, value : Object, startIndex : int) : int;

Description

Searches for the specified **System.Object** and returns the index of the last occurrence within the section of the one-dimensional **System.Array** that extends from the first element to the specified index.

Return Value: The index of the last occurrence of *value* within the section of *array* that extends from the first element to *startIndex* , if found; otherwise, the lower bound of the array - 1.

The one-dimensional **System.Array** is searched backward starting at *startIndex* and ending at the first element. The elements are compared to the specified value using the **System.Object.Equals(System.Object)** method. The one-dimensional **System.Array** to search. The **System.Object** to locate in *array*. The starting index of the backward search.

LastIndexOf

```

1
2 [C#] public static int LastIndexOf(Array array, object value, int startIndex, int
3 count);
4 [C++] public: static int LastIndexOf(Array* array, Object* value, int startIndex,
5 int count);
6 [VB] Public Shared Function LastIndexOf(ByVal array As Array, ByVal value As
7 Object, ByVal startIndex As Integer, ByVal count As Integer) As Integer
8 [JScript] public static function LastIndexOf(array : Array, value : Object,
9 startIndex : int, count : int) : int;
10

```

11 *Description*

12 Searches for the specified **System.Object** and returns the index of the last
13 occurrence within the section of the one-dimensional **System.Array** that contains
14 the specified number of elements and ends at the specified index.

15 *Return Value:* The index of the last occurrence of *value* within the section of *array*
16 that contains *count* number of elements and ends at *startIndex* , if found;
17 otherwise, the lower bound of the array - 1.

18 The one-dimensional **System.Array** is searched backward starting at
19 *startIndex* and ending at *startIndex - count + 1*. The elements are compared to the
20 specified value using the **System.Object.Equals(System.Object)** method. The
21 one-dimensional **System.Array** to search. The **System.Object** to locate in *array*.
22 The starting index of the backward search. The number of elements in the section
23 to search.

24 Reverse


```

1
2 [C#] public static void Reverse(Array array);
3 [C++] public: static void Reverse(Array* array);
4 [VB] Public Shared Sub Reverse(ByVal array As Array)
5 [JScript] public static function Reverse(array : Array); Reverses the order of the
6 elements in a one-dimensional System.Array or in a portion of the System.Array
7 .
8

```

Description

Reverses the sequence of the elements in the entire one-dimensional **System.Array** .

After a call to this method, the element at myArray[i] , where *i* is any index in the array, moves to myArray[j] , where *j* equals (myArray.Length + myArray.GetLowerBound(0)) - (i - myArray.GetLowerBound(0)) - 1 . The one-dimensional **System.Array** to reverse.

Reverse

```

17
18 [C#] public static void Reverse(Array array, int index, int length);
19 [C++] public: static void Reverse(Array* array, int index, int length);
20 [VB] Public Shared Sub Reverse(ByVal array As Array, ByVal index As Integer,
21 ByVal length As Integer)
22 [JScript] public static function Reverse(array : Array, index : int, length : int);
23

```

Description

Reverses the sequence of the elements in a section of the one-dimensional **System.Array** .

After a call to this method, the element at `myArray[i]` , where *i* is any index in the array, moves to `myArray[j]` , where *j* equals (`myArray.Length + myArray.GetLowerBound(0)) - (i - myArray.GetLowerBound(0)) - 1` . The one-dimensional **System.Array** to reverse. The starting index of the section to reverse. The number of elements in the section to reverse.

SetValue

[C#] public void SetValue(object value, int index);

[C++] public: void SetValue(Object* value, int index);

[VB] Public Sub SetValue(ByVal value As Object, ByVal index As Integer)

[JScript] public function SetValue(value : Object, index : int); Sets the specified **System.Array** elements to the specified value.

Description

Sets a value to the element at the specified position in a one-dimensional **System.Array** .

The **System.Array.GetLowerBound(System.Int32)** and **System.Array.GetUpperBound(System.Int32)** methods can determine whether the value of *index* is out of bounds. The new value for the specified element. The position of the **System.Array** element to set.

SetValue

[C#] public void SetValue(object value, int[] indices);

1 [C++] public: void SetValue(Object* value, int indices __gc[]);

2 [VB] Public Sub SetValue(ByVal value As Object, ByVal indices() As Integer)

3 [JScript] public function SetValue(value : Object, indices : int[]);

4
5 *Description*

6 Sets a value to the element at the specified position in a multidimensional

7 **System.Array** .

8 The number of elements in *indices* must equal the number of dimensions in
9 the **System.Array** . All elements in the *indices* array must collectively specify the
10 position of the desired element in the multidimensional **System.Array** . The new
11 value for the specified element. A one-dimensional array of indexes that specifies
12 the position of the element to set.

13 **SetValue**

14
15 [C#] public void SetValue(object value, int index1, int index2);

16 [C++] public: void SetValue(Object* value, int index1, int index2);

17 [VB] Public Sub SetValue(ByVal value As Object, ByVal index1 As Integer,
18 ByVal index2 As Integer)

19 [JScript] public function SetValue(value : Object, index1 : int, index2 : int);

20
21 *Description*

22 Sets a value to the element at the specified position in a two-dimensional

23 **System.Array** .

24 The **System.Array.GetLowerBound(System.Int32)** and

25 **System.Array.GetUpperBound(System.Int32)** methods can determine whether

any of the indexes is out of bounds. The new value for the specified element. The first-dimension index of the **System.Array** element to set. The second-dimension index of the **System.Array** element to set.

SetValue

[C#] public void SetValue(object value, int index1, int index2, int index3);
[C++] public: void SetValue(Object* value, int index1, int index2, int index3);
[VB] Public Sub SetValue(ByVal value As Object, ByVal index1 As Integer, ByVal index2 As Integer, ByVal index3 As Integer)
[JScript] public function SetValue(value : Object, index1 : int, index2 : int, index3 : int);

Description

Sets a value to the element at the specified position in a three-dimensional **System.Array**.

The **System.Array.GetLowerBound(System.Int32)** and **System.Array.GetUpperBound(System.Int32)** methods can determine whether any of the indexes is out of bounds. The new value for the specified element. The first-dimension index of the **System.Array** element to set. The second-dimension index of the **System.Array** element to set. The third-dimension index of the **System.Array** element to set.

Sort

[C#] public static void Sort(Array array);
[C++] public: static void Sort(Array* array);

1 [VB] Public Shared Sub Sort(ByVal array As Array)

2 [JScript] public static function Sort(array : Array); Sorts the elements in one-
3 dimensional **System.Array** objects.

4
5 *Description*

6 Sorts the elements in an entire one-dimensional **System.Array** using the
7 **System.IComparable** interface implemented by each element of the
8 **System.Array** .

9 Each element of *array* must implement the **System.IComparable** interface
10 to be capable of comparisons with every other element in *array* . The one-
11 dimensional **System.Array** to sort.

12 *Sort*

13
14 [C#] public static void Sort(Array keys, Array items);

15 [C++] public: static void Sort(Array* keys, Array* items);

16 [VB] Public Shared Sub Sort(ByVal keys As Array, ByVal items As Array)

17 [JScript] public static function Sort(keys : Array, items : Array);

18
19 *Description*

20 Sorts a pair of one-dimensional **System.Array** objects (one contains the
21 keys and the other contains the corresponding items) based on the keys in the first
22 **System.Array** using the **System.IComparable** interface implemented by each
23 key.

24 Each key in the *keys***System.Array** has a corresponding item in the
25 *items***System.Array** . When a key is repositioned during the sorting, the

corresponding item in the *itemsSystem.Array* is similarly repositioned. Therefore, the *itemsSystem.Array* is sorted according to the arrangement of the corresponding keys in the *keysSystem.Array*. The one-dimensional **System.Array** that contains the keys to sort. The one-dimensional **System.Array** that contains the items that correspond to each of the keys in the *keysSystem.Array*.

Sort

[C#] public static void Sort(Array array, IComparer comparer);

[C++] public: static void Sort(Array* array, IComparer* comparer);

[VB] Public Shared Sub Sort(ByVal array As Array, ByVal comparer As IComparer)

[JScript] public static function Sort(array : Array, comparer : IComparer);

Description

Sorts the elements in a one-dimensional **System.Array** using the specified **System.Collections.IComparer** interface.

If *comparer* is **null**, each element of *array* must implement the **System.IComparable** interface to be capable of comparisons with every other element in *array*. The one-dimensional **System.Array** to sort. The **System.Collections.IComparer** implementation to use when comparing elements.

Sort

[C#] public static void Sort(Array keys, Array items, IComparer comparer);

```

1 [C++] public: static void Sort(Array* keys, Array* items, IComparer* comparer);
2 [VB] Public Shared Sub Sort(ByVal keys As Array, ByVal items As Array, ByVal
3 comparer As IComparer)
4 [JScript] public static function Sort(keys : Array, items : Array, comparer :
5 IComparer);
6

```

Description

Sorts a pair of one-dimensional **System.Array** objects (one contains the keys and the other contains the corresponding items) based on the keys in the first **System.Array** using the specified **System.Collections.IComparer** interface.

Each key in the *keysSystem.Array* has a corresponding item in the *itemsSystem.Array* . When a key is repositioned during the sorting, the corresponding item in the *itemsSystem.Array* is similarly repositioned. Therefore, the *itemsSystem.Array* is sorted according to the arrangement of the corresponding keys in the *keysSystem.Array* . The one-dimensional **System.Array** that contains the keys to sort. The one-dimensional **System.Array** that contains the items that correspond to each of the keys in the *keysSystem.Array*. The **System.Collections.IComparer** implementation to use when comparing elements.

Sort

```

21
22 [C#] public static void Sort(Array array, int index, int length);
23 [C++] public: static void Sort(Array* array, int index, int length);
24 [VB] Public Shared Sub Sort(ByVal array As Array, ByVal index As Integer,
25 ByVal length As Integer)

```

1 [JScript] public static function Sort(array : Array, index : int, length : int);

3 *Description*

4 Sorts the elements in a section of a one-dimensional **System.Array** using
5 the **System.IComparable** interface implemented by each element of the
6 **System.Array** .

7 Each element within the specified section of *array* must implement the
8 **System.IComparable** interface to be capable of comparisons with every other
9 element in *array* . The one-dimensional **System.Array** to sort. The starting index
10 of the range to sort. The number of elements in the range to sort.

11 *Sort*

13 [C#] public static void Sort(Array keys, Array items, int index, int length);

14 [C++] public: static void Sort(Array* keys, Array* items, int index, int length);

15 [VB] Public Shared Sub Sort(ByVal keys As Array, ByVal items As Array, ByVal
16 index As Integer, ByVal length As Integer)

17 [JScript] public static function Sort(keys : Array, items : Array, index : int, length :
18 int);

20 *Description*

21 Sorts a section of a pair of one-dimensional **System.Array** objects (one
22 contains the keys and the other contains the corresponding items) based on the
23 keys in the first **System.Array** using the **System.IComparable** interface
24 implemented by each key.

Each key in the *keysSystem.Array* has a corresponding item in the *itemsSystem.Array* . When a key is repositioned during the sorting, the corresponding item in the *itemsSystem.Array* is similarly repositioned. Therefore, the *itemsSystem.Array* is sorted according to the arrangement of the corresponding keys in the *keysSystem.Array* . The one-dimensional **System.Array** that contains the keys to sort. The one-dimensional **System.Array** that contains the items that correspond to each of the keys in the *keysSystem.Array*. The starting index of the range to sort. The number of elements in the range to sort.

Sort

[C#] public static void Sort(Array array, int index, int length, IComparer comparer);

[C++] public: static void Sort(Array* array, int index, int length, IComparer* comparer);

[VB] Public Shared Sub Sort(ByVal array As Array, ByVal index As Integer, ByVal length As Integer, ByVal comparer As IComparer)

[JScript] public static function Sort(array : Array, index : int, length : int, comparer : IComparer);

Description

Sorts the elements in a section of a one-dimensional **System.Array** using the specified **System.Collections.IComparer** interface.

If *comparer* is **null** , each element within the specified section of *array* must implement the **System.IComparable** interface to be capable of comparisons

with every other element in *array* . The one-dimensional **System.Array** to sort.
The starting index of the range to sort. The number of elements in the range to
sort. The **System.Collections.IComparer** implementation to use when comparing
elements.

Sort

[C#] public static void Sort(Array keys, Array items, int index, int length,
IComparer comparer);
[C++] public: static void Sort(Array* keys, Array* items, int index, int length,
IComparer* comparer);
[VB] Public Shared Sub Sort(ByVal keys As Array, ByVal items As Array, ByVal
index As Integer, ByVal length As Integer, ByVal comparer As IComparer)
[JScript] public static function Sort(keys : Array, items : Array, index : int, length :
int, comparer : IComparer);

Description

Sorts a section of a pair of one-dimensional **System.Array** objects (one
contains the keys and the other contains the corresponding items) based on the
keys in the first **System.Array** using the specified
System.Collections.IComparer interface.

Each key in the *keysSystem.Array* has a corresponding item in the
itemsSystem.Array . When a key is repositioned during the sorting, the
corresponding item in the *itemsSystem.Array* is similarly repositioned. Therefore,
the *itemsSystem.Array* is sorted according to the arrangement of the
corresponding keys in the *keysSystem.Array* . The one-dimensional

System.Array that contains the keys to sort. The one-dimensional **System.Array** that contains the items that correspond to each of the keys in the *keysSystem.Array*. The starting index of the range to sort. The number of elements in the range to sort. The **System.Collections.IComparer** implementation to use when comparing elements.

IList.Add

[C#] int IList.Add(object value);

[C++] int IList::Add(Object* value);

[VB] Function Add(ByVal value As Object) As Integer Implements IList.Add

[JScript] function IList.Add(value : Object) : int;

IList.Clear

[C#] void IList.Clear();

[C++] void IList::Clear();

[VB] Sub Clear() Implements IList.Clear

[JScript] function IList.Clear();

IList.Contains

[C#] bool IList.Contains(object value);

[C++] bool IList::Contains(Object* value);

[VB] Function Contains(ByVal value As Object) As Boolean Implements

IList.Contains

[JScript] function IList.Contains(value : Object) : Boolean;

IList.IndexOf

```

1
2 [C#] int IList.IndexOf(object value);
3 [C++] int IList::IndexOf(Object* value);
4 [VB] Function IndexOf(ByVal value As Object) As Integer Implements
5 IList.IndexOf
6 [JScript] function IList.IndexOf(value : Object) : int;
7     IList.Insert
8
9 [C#] void IList.Insert(int index, object value);
10 [C++] void IList::Insert(int index, Object* value);
11 [VB] Sub Insert(ByVal index As Integer, ByVal value As Object) Implements
12 IList.Insert
13 [JScript] function IList.Insert(index : int, value : Object);
14     IList.Remove
15
16 [C#] void IList.Remove(object value);
17 [C++] void IList::Remove(Object* value);
18 [VB] Sub Remove(ByVal value As Object) Implements IList.Remove
19 [JScript] function IList.Remove(value : Object);
20     IList.RemoveAt
21
22 [C#] void IList.RemoveAt(int index);
23 [C++] void IList::RemoveAt(int index);
24 [VB] Sub RemoveAt(ByVal index As Integer) Implements IList.RemoveAt
25 [JScript] function IList.RemoveAt(index : int);

```

1 ArrayTypeMismatchException class (System)

2 ToString

3
4
5 *Description*

6 The exception that is thrown when an attempt is made to store an element
7 of the wrong type within an array.

8 **System.ArrayTypeMismatchException** uses the HRESULT
9 COR_E_ARRAYTYPEMISMATCH, which has the value 0x80131503.

10 ArrayTypeMismatchException

11 *Example Syntax:*

12 ToString

13
14 [C#] public ArrayTypeMismatchException();

15 [C++] public: ArrayTypeMismatchException();

16 [VB] Public Sub New()

17 [JScript] public function ArrayTypeMismatchException(); Initializes a new
18 instance of the **System.ArrayTypeMismatchException** class.

19
20 *Description*

21 Initializes a new instance of the **System.ArrayTypeMismatchException**
22 class with default properties.

23 The following table shows the initial property values for an instance of
24 **System.ArrayTypeMismatchException** .

25 ArrayTypeMismatchException

Example Syntax:

`ToString`

```
[C#] public ArrayTypeMismatchException(string message);  
[C++] public: ArrayTypeMismatchException(String* message);  
[VB] Public Sub New(ByVal message As String)  
[JScript] public function ArrayTypeMismatchException(message : String);
```

Description

Initializes a new instance of the **System.ArrayTypeMismatchException** class with a specified error message.

The following table shows the initial property values for an instance of **System.ArrayTypeMismatchException**. The error message that explains the reason for the exception.

`ArrayTypeMismatchException`

Example Syntax:

`ToString`

```
[C#] protected ArrayTypeMismatchException(SerializationInfo info,  
StreamingContext context);  
[C++] protected: ArrayTypeMismatchException(SerializationInfo* info,  
StreamingContext context);  
[VB] Protected Sub New(ByVal info As SerializationInfo, ByVal context As  
StreamingContext)  
[JScript] protected function ArrayTypeMismatchException(info :
```

1 SerializationInfo, context : StreamingContext);

2
3 *Description*

4 Initializes a new instance of the **System.ArrayTypeMismatchException**
5 class with serialized data.

6 This constructor is called during deserialization to reconstitute the
7 exception object transmitted over a stream. For more information, see . The object
8 that holds the serialized object data. The contextual information about the source
9 or destination.

10 ArrayTypeMismatchException

11 *Example Syntax:*

12 ToString

13
14 [C#] public ArrayTypeMismatchException(string message, Exception
15 innerException);

16 [C++] public: ArrayTypeMismatchException(String* message, Exception*
17 innerException);

18 [VB] Public Sub New(ByVal message As String, ByVal innerException As
19 Exception)

20 [JScript] public function ArrayTypeMismatchException(message : String,
21 innerException : Exception);

22
23 *Description*
24
25

1 Initializes a new instance of the **System.ArrayTypeMismatchException**
2 class with a specified error message and a reference to the inner exception that is
3 the root cause of this exception.

4 When an **Exception** *X* is thrown as a direct result of a previous exception *Y*,
5 the **System.Exception.InnerException** property of *X* should contain a reference
6 to *Y*. The **InnerException** property returns the same value as was passed into the
7 constructor, or **null** if the inner exception value was not supplied to the
8 constructor. The error message that explains the reason for the exception. An
9 instance of **System.Exception** that is the cause of the current **Exception**. If
10 *innerException* is non-null, then the current **Exception** is raised in a catch block
11 handling *innerException*.

12 HelpLink

13 HResult

14 InnerException

15 Message

16 Source

17 StackTrace

18 TargetSite

19 AssemblyLoadEventArgs class (System)

20 ToString

21
22
23 *Description*

24 Provides data for the **System.AppDomain.AssemblyLoad** event.

25 AssemblyLoadEventArgs

Example Syntax:

ToString

```
[C#] public AssemblyLoadEventArgs(Assembly loadedAssembly);  
[C++] public: AssemblyLoadEventArgs(Assembly* loadedAssembly);  
[VB] Public Sub New(ByVal loadedAssembly As Assembly)  
[JScript] public function AssemblyLoadEventArgs(loadedAssembly : Assembly);
```

Description

Initializes a new instance of the **System.AssemblyLoadEventArgs.AssemblyLoadEventArgs** class, using the specified **System.Reflection.Assembly** object. An instance that represents the currently loaded assembly.

LoadedAssembly

ToString

```
[C#] public Assembly LoadedAssembly {get;}  
[C++] public: __property Assembly* get_LoadedAssembly();  
[VB] Public ReadOnly Property LoadedAssembly As Assembly  
[JScript] public function get LoadedAssembly() : Assembly;
```

Description

Gets an **System.Reflection.Assembly** object that represents the currently loaded assembly.

AssemblyLoadEventHandler delegate (System)

ToString

Description

Represents the method that will handle the **System.AppDomain.AssemblyLoad** event of an **System.AppDomain** . The source of the event. An **System.AssemblyLoadEventArgs** that contains the event data.

AsyncCallback delegate (System)

ToString

Description

References the callback method to be called when the asynchronous operation is completed. The result of the asynchronous operation.

System.AsyncCallback provides a way for client applications to complete an asynchronous operation. This callback delegate is supplied to the client when the asynchronous operation is initiated. The event handler referenced by **System.AsyncCallback** contains program logic to finish processing the asynchronous task for the client.

Attribute class (System)

ToString

Description

Base class for custom attributes.

The **Attribute** class contains convenience methods to access and test custom attributes. While any user-defined type can be used as an attribute, it is expected that most attributes will be instances of types derived from **Attribute**.

Attribute

Example Syntax:

ToString

[C#] protected Attribute();

[C++] protected: Attribute();

[VB] Protected Sub New()

[JScript] protected function Attribute();

Description

Initializes a new instance of the **Attribute** class.

TypeId

ToString

[C#] public virtual object TypeId {get;}

[C++] public: __property virtual Object* get_TypeId();

[VB] Overridable Public ReadOnly Property TypeId As Object

[JScript] public function get TypeId() : Object;

Description

When implemented in a derived class, gets a unique identifier for this

Attribute .

As implemented, this identifier is merely the **System.Type** of the attribute.

However, it is intended that the unique identifier be used to identify two attributes of the same type.

Equals

[C#] public override bool Equals(object obj);

[C++] public: bool Equals(Object* obj);

[VB] Overrides Public Function Equals(ByVal obj As Object) As Boolean

[JScript] public override function Equals(obj : Object) : Boolean;

Description

Returns a value indicating whether this instance is equal to a specified object.

Return Value: **true** if *obj* equals the type and value of this instance; otherwise, **false** . An **System.Object** to compare with this instance or **null**.

GetCustomAttribute

[C#] public static Attribute GetCustomAttribute(Assembly element, Type attributeType);

[C++] public: static Attribute* GetCustomAttribute(Assembly* element, Type* attributeType);

[VB] Public Shared Function GetCustomAttribute(ByVal element As Assembly, ByVal attributeType As Type) As Attribute

1 [JScript] public static function GetCustomAttribute(element : Assembly,
2 attributeType : Type) : Attribute;

3
4 *Description*

5 Retrieves a custom attribute of a specified type applied to a specified
6 assembly and its ancestors.

7 *Return Value:* **null** , if no custom attribute of type *attributeType* is applied to
8 *element* . An object derived from class **System.Reflection.Assembly** that
9 describes a reusable, versionable, collection of modules. The **System.Type** object
10 to which the custom attributes are applied.

11 GetCustomAttribute

12
13 [C#] public static Attribute GetCustomAttribute(MemberInfo element, Type
14 attributeType);

15 [C++] public: static Attribute* GetCustomAttribute(MemberInfo* element, Type*
16 attributeType);

17 [VB] Public Shared Function GetCustomAttribute(ByVal element As
18 MemberInfo, ByVal attributeType As Type) As Attribute

19 [JScript] public static function GetCustomAttribute(element : MemberInfo,
20 attributeType : Type) : Attribute;

21
22 *Description*

23 Retrieves a custom attribute of a specified type applied to a specified
24 member of a class and its ancestors.

25 *Return Value:* **null** , if no custom attribute of type *attributeType* is applied to

element . An object derived from class **System.Reflection.MemberInfo** that describes a constructor, event, field, method, or property member of a class. The **System.Type** object to which the custom attributes are applied.

GetCustomAttribute

[C#] public static Attribute GetCustomAttribute(Module element, Type attributeType);

[C++] public: static Attribute* GetCustomAttribute(Module* element, Type* attributeType);

[VB] Public Shared Function GetCustomAttribute(ByVal element As Module, ByVal attributeType As Type) As Attribute

[JScript] public static function GetCustomAttribute(element : Module, attributeType : Type) : Attribute;

Description

Retrieves a custom attribute of a specified type applied to a specified module and its ancestors.

Return Value: **null** , if no custom attribute of type *attributeType* is applied to *element* . An object derived from class **System.Reflection.Module** that describes a portable executable file. The **System.Type** object to which the custom attributes are applied.

GetCustomAttribute

[C#] public static Attribute GetCustomAttribute(ParameterInfo element, Type attributeType);

1 [C++] public: static Attribute* GetCustomAttribute(ParameterInfo* element,
2 Type* attributeType);

3 [VB] Public Shared Function GetCustomAttribute(ByVal element As
4 ParameterInfo, ByVal attributeType As Type) As Attribute

5 [JScript] public static function GetCustomAttribute(element : ParameterInfo,
6 attributeType : Type) : Attribute;

7 8 *Description*

9 Retrieves a custom attribute of a specified type applied to a specified
10 parameter of a member of a class and its ancestors.

11 *Return Value:* **null** , if no custom attribute of type *attributeType* is applied to
12 *element* . An object derived from class **System.Reflection.ParameterInfo** that
13 describes a parameter of a member of a class. The **System.Type** object to which
14 the custom attributes are applied.

15 GetCustomAttribute

16
17 [C#] public static Attribute GetCustomAttribute(Assembly element, Type
18 attributeType, bool inherit);

19 [C++] public: static Attribute* GetCustomAttribute(Assembly* element, Type*
20 attributeType, bool inherit);

21 [VB] Public Shared Function GetCustomAttribute(ByVal element As Assembly,
22 ByVal attributeType As Type, ByVal inherit As Boolean) As Attribute

23 [JScript] public static function GetCustomAttribute(element : Assembly,
24 attributeType : Type, inherit : Boolean) : Attribute;

Description

Retrieves a custom attribute of a specified type applied to a specified assembly and optionally its ancestors.

Return Value: **null** , if no custom attribute of type *attributeType* is applied to *element* . An object derived from class **System.Reflection.Assembly** that describes a reusable, versionable, collection of modules. The **System.Type** object to which the custom attributes are applied. If **true**, specifies to also search the ancestors of *element* for custom attributes.

GetCustomAttribute

[C#] public static Attribute GetCustomAttribute(MemberInfo element, Type attributeType, bool inherit);

[C++] public: static Attribute* GetCustomAttribute(MemberInfo* element, Type* attributeType, bool inherit);

[VB] Public Shared Function GetCustomAttribute(ByVal element As MemberInfo, ByVal attributeType As Type, ByVal inherit As Boolean) As Attribute

[JScript] public static function GetCustomAttribute(element : MemberInfo, attributeType : Type, inherit : Boolean) : Attribute; Retrieves a custom attribute of a specified type applied to a specified member of a class.

Description

Retrieves a custom attribute of a specified type applied to a specified member of a class and optionally its ancestors.

1 *Return Value:* **null** , if no custom attribute of type *attributeType* is applied to
2 *element* . An object derived from class **System.Reflection.MemberInfo** that
3 describes a constructor, event, field, method, or property member of a class. The
4 **System.Type** object to which the custom attributes are applied. If **true**, specifies
5 to also search the ancestors of *element* for custom attributes.

6 GetCustomAttribute

7
8 [C#] public static Attribute GetCustomAttribute(Module element, Type
9 attributeType, bool inherit);

10 [C++] public: static Attribute* GetCustomAttribute(Module* element, Type*
11 attributeType, bool inherit);

12 [VB] Public Shared Function GetCustomAttribute(ByVal element As Module,
13 ByVal attributeType As Type, ByVal inherit As Boolean) As Attribute

14 [JScript] public static function GetCustomAttribute(element : Module,
15 attributeType : Type, inherit : Boolean) : Attribute;

17 Description

18 Retrieves a custom attribute of a specified type applied to a specified
19 module and optionally its ancestors.

20 *Return Value:* **null** , if no custom attribute of type *attributeType* is applied to
21 *element* . An object derived from class **System.Reflection.Module** that describes
22 a portable executable file. The **System.Type** object to which the custom attributes
23 are applied. If **true**, specifies to also search the ancestors of *element* for custom
24 attributes.

25 GetCustomAttribute

1
2 [C#] public static Attribute GetCustomAttribute(ParameterInfo element, Type
3 attributeType, bool inherit);

4 [C++] public: static Attribute* GetCustomAttribute(ParameterInfo* element,
5 Type* attributeType, bool inherit);

6 [VB] Public Shared Function GetCustomAttribute(ByVal element As
7 ParameterInfo, ByVal attributeType As Type, ByVal inherit As Boolean) As
8 Attribute

9 [JScript] public static function GetCustomAttribute(element : ParameterInfo,
10 attributeType : Type, inherit : Boolean) : Attribute;

11 12 *Description*

13 Retrieves a custom attribute of a specified type applied to a specified
14 parameter of a member of a class and optionally its ancestors.

15 *Return Value:* **null** , if no custom attribute of type *attributeType* is applied to
16 *element* . An object derived from class **System.Reflection.ParameterInfo** that
17 describes a parameter of a member of a class. The **System.Type** object to which
18 the custom attributes are applied. If **true**, specifies to also search the ancestors of
19 *element* for custom attributes.

20 GetCustomAttributes

21
22 [C#] public static Attribute[] GetCustomAttributes(Assembly element);

23 [C++] public: static Attribute* GetCustomAttributes(Assembly* element) [];

24 [VB] Public Shared Function GetCustomAttributes(ByVal element As Assembly)
25 As Attribute()

1 [JScript] public static function GetCustomAttributes(element : Assembly) :

2 Attribute[];

3
4 *Description*

5 Retrieves an array of the custom attributes of a specified type applied to a
6 specified assembly and its ancestors.

7 *Return Value:* An **System.Attribute** array containing the custom attributes applied
8 to *element*. -or- An empty array if no such custom attributes exist.

9 Return value contains the custom attributes for ancestors of *element*. An
10 object derived from class **System.Reflection.Assembly** that describes a reusable,
11 versionable, collection of modules.

12 GetCustomAttributes

13
14 [C#] public static Attribute[] GetCustomAttributes(MemberInfo element);

15 [C++] public: static Attribute* GetCustomAttributes(MemberInfo* element) [];

16 [VB] Public Shared Function GetCustomAttributes(ByVal element As
17 MemberInfo) As Attribute()

18 [JScript] public static function GetCustomAttributes(element : MemberInfo) :

19 Attribute[];

20
21 *Description*

22 Retrieves an array of the custom attributes applied to a specified member of
23 a class and its ancestors.

24 *Return Value:* An **System.Attribute** array containing the custom attributes applied
25 to *element*. -or- An empty array if no such custom attributes exist.

Return value contains the custom attributes for ancestors of *element* . An object derived from class **System.Reflection.MemberInfo** that describes a constructor, event, field, method, or property member of a class.

GetCustomAttributes

[C#] public static Attribute[] GetCustomAttributes(Module element);

[C++] public: static Attribute* GetCustomAttributes(Module* element) [];

[VB] Public Shared Function GetCustomAttributes(ByVal element As Module)

As Attribute()

[JScript] public static function GetCustomAttributes(element : Module) :

Attribute[];

Description

Retrieves an array of the custom attributes of a specified type applied to a specified module and its ancestors.

Return Value: An **System.Attribute** array containing the custom attributes applied to *element*. -or- An empty array if no such custom attributes exist.

Return value contains the custom attributes for ancestors of *element* . An object derived from class **System.Reflection.Module** that describes a portable executable file.

GetCustomAttributes

[C#] public static Attribute[] GetCustomAttributes(ParameterInfo element);

[C++] public: static Attribute* GetCustomAttributes(ParameterInfo* element) [];

[VB] Public Shared Function GetCustomAttributes(ByVal element As

ParameterInfo) As Attribute()

[JScript] public static function GetCustomAttributes(element : ParameterInfo) :
Attribute[];

Description

Retrieves an array of the custom attributes of a specified type applied to a specified parameter of a member of a class and its ancestors.

Return Value: An **System.Attribute** array containing the custom attributes applied to *element*. -or- An empty array if no such custom attributes exist.

Return value contains the custom attributes for ancestors of *element*. An object derived from class **System.Reflection.ParameterInfo** that describes a parameter of a member of a class.

GetCustomAttributes

[C#] public static Attribute[] GetCustomAttributes(Assembly element, bool inherit);

[C++] public: static Attribute* GetCustomAttributes(Assembly* element, bool inherit) [];

[VB] Public Shared Function GetCustomAttributes(ByVal element As Assembly, ByVal inherit As Boolean) As Attribute()

[JScript] public static function GetCustomAttributes(element : Assembly, inherit : Boolean) : Attribute[];

Description

Retrieves an array of the custom attributes of a specified type applied to a specified assembly and optionally its ancestors.

Return Value: An **System.Attribute** array containing the custom attributes applied to *element*. -or- An empty array if no such custom attributes exist.

Return value contains the custom attributes for ancestors of *element* if *inherit* is **true**. An object derived from class **System.Reflection.Assembly** that describes a reusable, versionable, collection of modules. If **true**, specifies to also search the ancestors of *element* for custom attributes.

GetCustomAttributes

[C#] public static Attribute[] GetCustomAttributes(Assembly element, Type attributeType);

[C++] public: static Attribute* GetCustomAttributes(Assembly* element, Type* attributeType) [];

[VB] Public Shared Function GetCustomAttributes(ByVal element As Assembly, ByVal attributeType As Type) As Attribute()

[JScript] public static function GetCustomAttributes(element : Assembly, attributeType : Type) : Attribute[];

Description

Retrieves an array of the custom attributes of a specified type applied to a specified assembly and its ancestors.

Return Value: An **System.Attribute** array containing the custom attributes of type *attributeType* applied to *element*. -or- An empty array if no such custom attributes exist.

Return value contains the custom attributes for ancestors of *element* . An object derived from class **System.Reflection.Assembly** that describes a reusable, versionable, collection of modules. The **System.Type** object to which the custom attributes are applied.

GetCustomAttributes

```
[C#] public static Attribute[] GetCustomAttributes(MemberInfo element, bool
inherit);
[C++] public: static Attribute* GetCustomAttributes(MemberInfo* element, bool
inherit) [];
[VB] Public Shared Function GetCustomAttributes(ByVal element As
MemberInfo, ByVal inherit As Boolean) As Attribute()
[JScript] public static function GetCustomAttributes(element : MemberInfo,
inherit : Boolean) : Attribute[];
```

Description

Retrieves an array of the custom attributes of a specified member of a class and its ancestors.

Return Value: An **System.Attribute** array containing the custom attributes applied to *element*. -or- An empty array if no such custom attributes exist.

Return value contains the custom attributes for ancestors of *element* if *inherit* is **true** . An object derived from class **System.Reflection.MemberInfo** that describes a constructor, event, field, method, or property member of a class. If **true**, specifies to also search the ancestors of *element* for custom attributes.

GetCustomAttributes

```

1
2 [C#] public static Attribute[] GetCustomAttributes(MemberInfo element, Type
3 type);
4 [C++] public: static Attribute* GetCustomAttributes(MemberInfo* element,
5 Type* type) [];
6 [VB] Public Shared Function GetCustomAttributes(ByVal element As
7 MemberInfo, ByVal type As Type) As Attribute()
8 [JScript] public static function GetCustomAttributes(element : MemberInfo, type :
9 Type) : Attribute[]; Retrieves an array of the custom attributes of a specified type
10 applied to a specified member of a class.
11

```

Description

Retrieves an array of the custom attributes of a specified type applied to a specified member of a class and its ancestors.

Return Value: An **System.Attribute** array containing the custom attributes of type *type* applied to *element*. -or- An empty array if no such custom attributes exist.

Return value contains the custom attributes for ancestors of *element*. An object derived from class **System.Reflection.MemberInfo** that describes a constructor, event, field, method, or property member of a class. The **System.Type** object to which the custom attributes are applied.

GetCustomAttributes

```

22
23 [C#] public static Attribute[] GetCustomAttributes(Module element, bool inherit);
24 [C++] public: static Attribute* GetCustomAttributes(Module* element, bool
25 inherit) [];

```



```

1 [VB] Public Shared Function GetCustomAttributes(ByVal element As Module,
2 ByVal inherit As Boolean) As Attribute()
3 [JScript] public static function GetCustomAttributes(element : Module, inherit :
4 Boolean) : Attribute[];

```

Description

Retrieves an array of the custom attributes of a specified type applied to a specified module and optionally its ancestors.

Return Value: An **System.Attribute** array containing the custom attributes applied to *element*. -or- An empty array if no such custom attributes exist.

Return value contains the custom attributes for ancestors of *element* if *inherit* is **true** . An object derived from class **System.Reflection.Module** that describes a portable executable file. If **true**, specifies to also search the ancestors of *element* for custom attributes.

GetCustomAttributes

```

17 [C#] public static Attribute[] GetCustomAttributes(Module element, Type
18 attributeType);
19 [C++] public: static Attribute* GetCustomAttributes(Module* element, Type*
20 attributeType) [];
21 [VB] Public Shared Function GetCustomAttributes(ByVal element As Module,
22 ByVal attributeType As Type) As Attribute()
23 [JScript] public static function GetCustomAttributes(element : Module,
24 attributeType : Type) : Attribute[];

```

Description

Retrieves an array of the custom attributes of a specified type applied to a specified module and its ancestors.

Return Value: An **System.Attribute** array containing the custom attributes of type *attributeType* applied to *element*. -or- An empty array if no such custom attributes exist.

Return value contains the custom attributes for ancestors of *element*. An object derived from class **System.Reflection.Module** that describes a portable executable file. The **System.Type** object to which the custom attributes are applied.

GetCustomAttributes

```
[C#] public static Attribute[] GetCustomAttributes(ParameterInfo element, bool inherit);
```

```
[C++] public: static Attribute* GetCustomAttributes(ParameterInfo* element, bool inherit) [];
```

```
[VB] Public Shared Function GetCustomAttributes(ByVal element As ParameterInfo, ByVal inherit As Boolean) As Attribute()
```

```
[JScript] public static function GetCustomAttributes(element : ParameterInfo, inherit : Boolean) : Attribute[];
```

Description

Retrieves an array of the custom attributes of a specified type applied to a specified parameter of a member of a class and optionally its ancestors.

1 *Return Value:* An **System.Attribute** array containing the custom attributes applied
2 to *element*. -or- An empty array if no such custom attributes exist.

3 Return value contains the custom attributes for ancestors of *element* if
4 *inherit* is **true** . An object derived from class **System.Reflection.ParameterInfo**
5 that describes a parameter of a member of a class. If **true**, specifies to also search
6 the ancestors of *element* for custom attributes.

7 GetCustomAttributes

8
9 [C#] public static Attribute[] GetCustomAttributes(ParameterInfo element, Type
10 attributeType);

11 [C++] public: static Attribute* GetCustomAttributes(ParameterInfo* element,
12 Type* attributeType) [];

13 [VB] Public Shared Function GetCustomAttributes(ByVal element As
14 ParameterInfo, ByVal attributeType As Type) As Attribute()

15 [JScript] public static function GetCustomAttributes(element : ParameterInfo,
16 attributeType : Type) : Attribute[];

17 18 *Description*

19 Retrieves an array of the custom attributes of a specified type applied to a
20 specified parameter of a member of a class and its ancestors.

21 *Return Value:* An **System.Attribute** array containing the custom attributes of type
22 *attributeType* applied to *element*. -or- An empty array if no such custom attributes
23 exist.

24 Return value contains the custom attributes for ancestors of *element* . An
25 object derived from class **System.Reflection.ParameterInfo** that describes a

parameter of a member of a class. The **System.Type** object to which the custom attributes are applied.

GetCustomAttributes

[C#] public static Attribute[] GetCustomAttributes(Assembly element, Type attributeType, bool inherit);

[C++] public: static Attribute* GetCustomAttributes(Assembly* element, Type* attributeType, bool inherit) [];

[VB] Public Shared Function GetCustomAttributes(ByVal element As Assembly, ByVal attributeType As Type, ByVal inherit As Boolean) As Attribute()

[JScript] public static function GetCustomAttributes(element : Assembly, attributeType : Type, inherit : Boolean) : Attribute[];

Description

Retrieves an array of the custom attributes of a specified type applied to a specified assembly and optionally its ancestors.

Return Value: An **System.Attribute** array containing the custom attributes of type *attributeType* applied to *element*. -or- An empty array if no such custom attributes exist.

Return value contains the custom attributes for ancestors of *element* if *inherit* is **true**. An object derived from class **System.Reflection.Assembly** that describes a reusable, versionable, collection of modules. The **System.Type** object to which the custom attributes are applied. If **true**, specifies to also search the ancestors of *element* for custom attributes.

GetCustomAttributes

```

1
2 [C#] public static Attribute[] GetCustomAttributes(MemberInfo element, Type
3 type, bool inherit);
4 [C++] public: static Attribute* GetCustomAttributes(MemberInfo* element,
5 Type* type, bool inherit) [];
6 [VB] Public Shared Function GetCustomAttributes(ByVal element As
7 MemberInfo, ByVal type As Type, ByVal inherit As Boolean) As Attribute()
8 [JScript] public static function GetCustomAttributes(element : MemberInfo, type :
9 Type, inherit : Boolean) : Attribute[];
10

```

Description

Retrieves an array of the custom attributes of a specified type applied to a specified member of a class and optionally its ancestors.

Return Value: An **System.Attribute** array containing the custom attributes of type *type* applied to *element*. -or- An empty array if no such custom attributes exist.

Return value contains the custom attributes for ancestors of *element* if *inherit* is **true**. An object derived from class **System.Reflection.MemberInfo** that describes a constructor, event, field, method, or property member of a class. The **System.Type** object to which the custom attributes are applied. If **true**, specifies to also search the ancestors of *element* for custom attributes.

GetCustomAttributes

```

22
23 [C#] public static Attribute[] GetCustomAttributes(Module element, Type
24 attributeType, bool inherit);
25 [C++] public: static Attribute* GetCustomAttributes(Module* element, Type*

```

1 attributeType, bool inherit) [];

2 [VB] Public Shared Function GetCustomAttributes(ByVal element As Module,
3 ByVal attributeType As Type, ByVal inherit As Boolean) As Attribute()
4 [JScript] public static function GetCustomAttributes(element : Module,
5 attributeType : Type, inherit : Boolean) : Attribute[];

7 *Description*

8 Retrieves an array of the custom attributes of a specified type applied to a
9 specified module and optionally its ancestors.

10 *Return Value:* An **System.Attribute** array containing the custom attributes of type
11 *attributeType* applied to *element*. -or- An empty array if no such custom attributes
12 exist.

13 Return value contains the custom attributes for ancestors of *element* if
14 *inherit* is **true** . An object derived from class **System.Reflection.Module** that
15 describes a portable executable file. The **System.Type** object to which the custom
16 attributes are applied. If **true**, specifies to also search the ancestors of *element* for
17 custom attributes.

18 GetCustomAttributes

19
20 [C#] public static Attribute[] GetCustomAttributes(ParameterInfo element, Type
21 attributeType, bool inherit);

22 [C++] public: static Attribute* GetCustomAttributes(ParameterInfo* element,
23 Type* attributeType, bool inherit) [];

24 [VB] Public Shared Function GetCustomAttributes(ByVal element As
25 ParameterInfo, ByVal attributeType As Type, ByVal inherit As Boolean) As

1 Attribute()

2 [JScript] public static function GetCustomAttributes(element : ParameterInfo,
3 attributeType : Type, inherit : Boolean) : Attribute[];

4
5 *Description*

6 Retrieves an array of the custom attributes of a specified type applied to a
7 specified parameter of a member of a class and optionally its ancestors.

8 *Return Value:* An **System.Attribute** array containing the custom attributes of type
9 *attributeType* applied to *element*. -or- An empty array if no such custom attributes
10 exist.

11 Return value contains the custom attributes for ancestors of *element* if
12 *inherit* is **true** . An object derived from class **System.Reflection.ParameterInfo**
13 that describes a parameter of a member of a class. The **System.Type** object to
14 which the custom attributes are applied. If **true**, specifies to also search the
15 ancestors of *element* for custom attributes.

16 GetHashCode

17
18 [C#] public override int GetHashCode();

19 [C++] public: int GetHashCode();

20 [VB] Overrides Public Function GetHashCode() As Integer

21 [JScript] public override function GetHashCode() : int;

22
23 *Description*

24 Returns the hash code for this instance.

25 *Return Value:* A 32-bit signed integer hash code.

IsDefaultAttribute

[C#] public virtual bool IsDefaultAttribute();
[C++] public: virtual bool IsDefaultAttribute();
[VB] Overridable Public Function IsDefaultAttribute() As Boolean
[JScript] public function IsDefaultAttribute() : Boolean;

Description

When overridden in a derived class, returns an indication whether the value of this instance is the default value for the derived class.

Return Value: **true** if this instance is the default attribute for the class; otherwise, **false**.

The default implementation of this class returns **false**, and must be implemented in the derived class to be useful to that class.

IsDefined

[C#] public static bool IsDefined(Assembly element, Type attributeType);
[C++] public: static bool IsDefined(Assembly* element, Type* attributeType);
[VB] Public Shared Function IsDefined(ByVal element As Assembly, ByVal attributeType As Type) As Boolean
[JScript] public static function IsDefined(element : Assembly, attributeType : Type) : Boolean;

Description

Determines whether any custom attributes of a specified type are applied to a specified assembly.

Return Value: **true** if a custom attribute of type *attributeType* is applied to *element*; otherwise, **false**.

The ancestors of *element* are not searched for custom attributes. An object derived from class **System.Reflection.Assembly** that describes a reusable, versionable, collection of modules. The **System.Type** object to which the custom attributes are applied.

IsDefined

[C#] public static bool IsDefined(MemberInfo element, Type attributeType);

[C++] public: static bool IsDefined(MemberInfo* element, Type* attributeType);

[VB] Public Shared Function IsDefined(ByVal element As MemberInfo, ByVal attributeType As Type) As Boolean

[JScript] public static function IsDefined(element : MemberInfo, attributeType : Type) : Boolean; Determines whether any custom attributes of a specified type are applied to a specified member of a class.

Description

Determines whether any custom attributes of a specified type are applied to a specified member of a class and its ancestors.

Return Value: **true** if a custom attribute of type *attributeType* is applied to *element*; otherwise, **false**.

The ancestors of *element* are searched for custom attributes if *element* is a method or a type. An object derived from class **System.Reflection.MemberInfo**

that describes a constructor, event, field, method, type, or property member of a class. The **System.Type** object to which the custom attributes are applied.

IsDefined

[C#] public static bool IsDefined(Module element, Type attributeType);

[C++] public: static bool IsDefined(Module* element, Type* attributeType);

[VB] Public Shared Function IsDefined(ByVal element As Module, ByVal attributeType As Type) As Boolean

[JScript] public static function IsDefined(element : Module, attributeType : Type) : Boolean;

Description

Determines whether any custom attributes of a specified type are applied to a specified module.

Return Value: **true** if a custom attribute of type *attributeType* is applied to *element*; otherwise, **false**.

The ancestors of *element* are not searched for custom attributes. An object derived from class **System.Reflection.Module** that describes a portable executable file. The **System.Type** object to which the custom attributes are applied.

IsDefined

[C#] public static bool IsDefined(ParameterInfo element, Type attributeType);

[C++] public: static bool IsDefined(ParameterInfo* element, Type* attributeType);

```

1 [VB] Public Shared Function IsDefined(ByVal element As ParameterInfo, ByVal
2 attributeType As Type) As Boolean
3 [JScript] public static function IsDefined(element : ParameterInfo, attributeType :
4 Type) : Boolean;

```

Description

Determines whether any custom attributes of a specified type are applied to a specified parameter of a member of a class and its ancestors.

Return Value: **true** if a custom attribute of type *attributeType* is applied to *element*; otherwise, **false**.

The ancestors of *element* are searched for custom attributes. An object derived from class **System.Reflection.ParameterInfo** that describes a parameter of a member of a class. The **System.Type** object to which the custom attributes are applied.

IsDefined

```

17 [C#] public static bool IsDefined(Assembly element, Type attributeType, bool
18 inherit);
19 [C++] public: static bool IsDefined(Assembly* element, Type* attributeType,
20 bool inherit);
21 [VB] Public Shared Function IsDefined(ByVal element As Assembly, ByVal
22 attributeType As Type, ByVal inherit As Boolean) As Boolean
23 [JScript] public static function IsDefined(element : Assembly, attributeType :
24 Type, inherit : Boolean) : Boolean;

```

Description

Determines whether any custom attributes of a specified type are applied to a specified assembly.

Return Value: **true** if a custom attribute of type *attributeType* is applied to *element*; otherwise, **false**.

This method ignores the value of parameter *inherit*. The ancestors of *element* are not searched for custom attributes. An object derived from class **System.Reflection.Assembly** that describes a reusable, versionable, collection of modules. The **System.Type** object to which the custom attributes are applied. If **true**, specifies to also search the ancestors of *element* for custom attributes.

IsDefined

[C#] public static bool IsDefined(MemberInfo element, Type attributeType, bool inherit);

[C++] public: static bool IsDefined(MemberInfo* element, Type* attributeType, bool inherit);

[VB] Public Shared Function IsDefined(ByVal element As MemberInfo, ByVal attributeType As Type, ByVal inherit As Boolean) As Boolean

[JScript] public static function IsDefined(element : MemberInfo, attributeType : Type, inherit : Boolean) : Boolean;

Description

Determines whether any custom attributes of a specified type are applied to a specified member of a class and optionally its ancestors.

1 *Return Value:* **true** if a custom attribute of type *attributeType* is applied to *element*
2 ; otherwise, **false** .

3 The ancestors of *element* are searched for custom attributes if *inherit* is **true**
4 and *element* is a method or a type. An object derived from class
5 **System.Reflection.MemberInfo** that describes a constructor, event, field,
6 method, type, or property member of a class. The **System.Type** object to which
7 the custom attributes are applied. If **true**, specifies to also search the ancestors of
8 *element* for custom attributes.

9 IsDefined

10
11 [C#] public static bool IsDefined(Module element, Type attributeType, bool
12 inherit);

13 [C++] public: static bool IsDefined(Module* element, Type* attributeType, bool
14 inherit);

15 [VB] Public Shared Function IsDefined(ByVal element As Module, ByVal
16 attributeType As Type, ByVal inherit As Boolean) As Boolean

17 [JScript] public static function IsDefined(element : Module, attributeType : Type,
18 inherit : Boolean) : Boolean;

19 20 *Description*

21 Determines whether any custom attributes of a specified type are applied to
22 a specified module.

23 *Return Value:* **true** if a custom attribute of type *attributeType* is applied to *element*
24 ; otherwise, **false** .

This method ignores the value of parameter *inherit* . The ancestors of *element* are not searched for custom attributes. An object derived from class **System.Reflection.Module** that describes a portable executable file. The **System.Type** object to which the custom attributes are applied. If **true**, specifies to also search the ancestors of *element* for custom attributes.

IsDefined

[C#] public static bool IsDefined(ParameterInfo element, Type attributeType, bool inherit);

[C++] public: static bool IsDefined(ParameterInfo* element, Type* attributeType, bool inherit);

[VB] Public Shared Function IsDefined(ByVal element As ParameterInfo, ByVal attributeType As Type, ByVal inherit As Boolean) As Boolean

[JScript] public static function IsDefined(element : ParameterInfo, attributeType : Type, inherit : Boolean) : Boolean;

Description

Determines whether any custom attributes of a specified type are applied to a specified parameter of a member of a class and optionally its ancestors.

Return Value: **true** if a custom attribute of type *attributeType* is applied to *element* ; otherwise, **false** .

The ancestors of *element* are searched for custom attributes if *inherit* is **true** and *element* is a method. An object derived from class **System.Reflection.ParameterInfo** that describes a parameter of a member of a

class. The **System.Type** object to which the custom attributes are applied. If **true**, specifies to also search the ancestors of *element* for custom attributes.

Match

[C#] public virtual bool Match(object obj);

[C++] public: virtual bool Match(Object* obj);

[VB] Overridable Public Function Match(ByVal obj As Object) As Boolean

[JScript] public function Match(obj : Object) : Boolean;

Description

When overridden in a derived class, returns a value indicating whether this instance equals a specified object.

Return Value: **true** if this instance equals *obj* ; otherwise, **false** .

This method determines if one **Attribute** equals another. Its default implementation is the same as **System.Attribute.Equals(System.Object)** , which performs a value and reference comparison. Override this method to implement support for attribute values, such as flags or bitfields, that consist of components that are meaningful in themselves. An **System.Object** to compare with this instance of **Attribute**.

AttributeTargets enumeration (System)

ToString

Description

Specifies the elements to which it is valid to apply an attribute.

AttributeTargets enumeration values can be combined with a bitwise OR operation to get the desired combination.

ToString

[C#] public const AttributeTargets All;

[C++] public: const AttributeTargets All;

[VB] Public Const All As AttributeTargets

[JScript] public var All : AttributeTargets;

Description

Attribute can be applied to any element.

ToString

[C#] public const AttributeTargets Assembly;

[C++] public: const AttributeTargets Assembly;

[VB] Public Const Assembly As AttributeTargets

[JScript] public var Assembly : AttributeTargets;

Description

Attribute can be applied to an assembly.

ToString

[C#] public const AttributeTargets Class;

[C++] public: const AttributeTargets Class;

[VB] Public Const Class As AttributeTargets

1 [JScript] public var Class : AttributeTargets;

3 *Description*

4 Attribute can be applied to a class.

5 ToString

7 [C#] public const AttributeTargets Constructor;

8 [C++] public: const AttributeTargets Constructor;

9 [VB] Public Const Constructor As AttributeTargets

10 [JScript] public var Constructor : AttributeTargets;

12 *Description*

13 Attribute can be applied to a constructor.

14 ToString

16 [C#] public const AttributeTargets Delegate;

17 [C++] public: const AttributeTargets Delegate;

18 [VB] Public Const Delegate As AttributeTargets

19 [JScript] public var Delegate : AttributeTargets;

21 *Description*

22 Attribute can be applied to a delegate.

23 ToString

25 [C#] public const AttributeTargets Enum;

1 [C++] public: const AttributeTargets Enum;
2 [VB] Public Const Enum As AttributeTargets
3 [JScript] public var Enum : AttributeTargets;
4

5 *Description*

6 Attribute can be applied to an enumeration.

7 ToString
8

9 [C#] public const AttributeTargets Event;
10 [C++] public: const AttributeTargets Event;
11 [VB] Public Const Event As AttributeTargets
12 [JScript] public var Event : AttributeTargets;
13

14 *Description*

15 Attribute can be applied to an event.

16 ToString
17

18 [C#] public const AttributeTargets Field;
19 [C++] public: const AttributeTargets Field;
20 [VB] Public Const Field As AttributeTargets
21 [JScript] public var Field : AttributeTargets;
22

23 *Description*

24 Attribute can be applied to a field.

25 ToString

1
2 [C#] public const AttributeTargets Interface;
3 [C++] public: const AttributeTargets Interface;
4 [VB] Public Const Interface As AttributeTargets
5 [JScript] public var Interface : AttributeTargets;

6
7 *Description*

8 Attribute can be applied to an interface.

9 ToString

10
11 [C#] public const AttributeTargets Method;
12 [C++] public: const AttributeTargets Method;
13 [VB] Public Const Method As AttributeTargets
14 [JScript] public var Method : AttributeTargets;

15
16 *Description*

17 Attribute can be applied to a method.

18 ToString

19
20 [C#] public const AttributeTargets Module;
21 [C++] public: const AttributeTargets Module;
22 [VB] Public Const Module As AttributeTargets
23 [JScript] public var Module : AttributeTargets;

24
25 *Description*

Attribute can be applied to a module.

ToString

[C#] public const AttributeTargets Parameter;

[C++] public: const AttributeTargets Parameter;

[VB] Public Const Parameter As AttributeTargets

[JScript] public var Parameter : AttributeTargets;

Description

Attribute can be applied to a parameter.

ToString

[C#] public const AttributeTargets Property;

[C++] public: const AttributeTargets Property;

[VB] Public Const Property As AttributeTargets

[JScript] public var Property : AttributeTargets;

Description

Attribute can be applied to a property.

ToString

[C#] public const AttributeTargets ReturnValue;

[C++] public: const AttributeTargets ReturnValue;

[VB] Public Const ReturnValue As AttributeTargets

[JScript] public var ReturnValue : AttributeTargets;

1
2 *Description*

3 Attribute can be applied to a Return value.

4 ToString

5
6 [C#] public const AttributeTargets Struct;

7 [C++] public: const AttributeTargets Struct;

8 [VB] Public Const Struct As AttributeTargets

9 [JScript] public var Struct : AttributeTargets;

10
11 *Description*

12 Attribute can be applied to a value type.

13 AttributeUsageAttribute class (System)

14 ToString

15
16
17 *Description*

18 Specifies the usage of another attribute class. This class cannot be inherited.

19 When you are defining your own attribute class, you can control the
20 manner in which it is used by placing an **System.AttributeUsageAttribute** on
21 your attribute class. The indicated attribute class must derive from
22 **System.Attribute** , either directly or indirectly.

23 AttributeUsageAttribute

24 *Example Syntax:*

25 ToString

```

1
2 [C#] public AttributeUsageAttribute(AttributeTargets validOn);
3 [C++] public: AttributeUsageAttribute(AttributeTargets validOn);
4 [VB] Public Sub New(ByVal validOn As AttributeTargets)
5 [JScript] public function AttributeUsageAttribute(validOn : AttributeTargets);
6

```

7 *Description*

8 Initializes a new instance of the **System.AttributeUsageAttribute** class
9 with the specified list of **System.AttributeTargets** , the
10 **System.AttributeUsageAttribute.AllowMultiple** value, and the
11 **System.AttributeUsageAttribute.Inherited** value.

12 You can combine several **System.AttributeTargets** values using a bitwise
13 OR operation to get the desired combination of valid program elements. The set of
14 values combined using a bitwise OR operation to indicate which program
15 elements are valid.

16 AllowMultiple

17 ToString

```

18
19 [C#] public bool AllowMultiple {get; set;}
20 [C++] public: __property bool get_ AllowMultiple();public: __property void
21 set_ AllowMultiple(bool);
22 [VB] Public Property AllowMultiple As Boolean
23 [JScript] public function get AllowMultiple() : Boolean;public function set
24 AllowMultiple(Boolean);
25

```

1
2 *Description*

3 Gets or sets a Boolean value indicating whether more than one instances of
4 the indicated attribute can be specified for a single program element.

5 An attribute that can be specified more than once for a program element is
6 called a multi-use attribute. An attribute that can be specified only once is called a
7 single-use attribute.

8 Inherited

9 ToString

10
11 [C#] public bool Inherited {get; set;}

12 [C++] public: __property bool get_Inherited();public: __property void
13 set_Inherited(bool);

14 [VB] Public Property Inherited As Boolean

15 [JScript] public function get Inherited() : Boolean;public function set
16 Inherited(Boolean);

17
18 *Description*

19 Gets or sets a Boolean value indicating whether the indicated attribute is
20 inherited by derived classes or overridden members.

21 TypeId

22 ValidOn

23 ToString

1
2
3 *Description*

4 Gets a set of values identifying which program elements that the indicated
5 attribute can be applied to.

6 **BadImageFormatException** class (System)

7 ToString
8
9

10 *Description*

11 The exception that is thrown when the file image of a DLL or an executable
12 program is invalid.

13 **System.BadImageFormatException** uses the HRESULT
14 COR_E_BADIMAGEFORMAT, which has the value 0x8007000B.

15 **BadImageFormatException**

16 *Example Syntax:*

17 ToString
18

19 [C#] public BadImageFormatException();

20 [C++] public: BadImageFormatException();

21 [VB] Public Sub New()

22 [JScript] public function BadImageFormatException(); Initializes a new instance
23 of the **System.BadImageFormatException** class.
24

25 *Description*

1 Initializes a new instance of the **System.BadImageFormatException** class
2 with default properties.

3 The following table shows the initial property values for an instance of
4 **System.BadImageFormatException** .

5 BadImageFormatException

6 *Example Syntax:*

7 ToString

9 [C#] public BadImageFormatException(string message);

10 [C++] public: BadImageFormatException(String* message);

11 [VB] Public Sub New(ByVal message As String)

12 [JScript] public function BadImageFormatException(message : String);

14 *Description*

15 Initializes a new instance of the **System.BadImageFormatException** class
16 with a specified error message.

17 The following table shows the initial property values for an instance of
18 **System.BadImageFormatException** . The error message that explains the reason
19 for the exception.

20 BadImageFormatException

21 *Example Syntax:*

22 ToString

24 [C#] protected BadImageFormatException(SerializationInfo info,
25 StreamingContext context);

```

1 [C++] protected: BadImageFormatException(SerializationInfo* info,
2 StreamingContext context);
3 [VB] Protected Sub New(ByVal info As SerializationInfo, ByVal context As
4 StreamingContext)
5 [JScript] protected function BadImageFormatException(info : SerializationInfo,
6 context : StreamingContext);
7

```

Description

Initializes a new instance of the **System.BadImageFormatException** class with serialized data.

This constructor is called during deserialization to reconstitute the exception object transmitted over a stream. For more information, see . The object that holds the serialized object data. The contextual information about the source or destination.

BadImageFormatException

Example Syntax:

ToString

```

19 [C#] public BadImageFormatException(string message, Exception inner);
20 [C++] public: BadImageFormatException(String* message, Exception* inner);
21 [VB] Public Sub New(ByVal message As String, ByVal inner As Exception)
22 [JScript] public function BadImageFormatException(message : String, inner :
23 Exception);
24

```

Description

1 Initializes a new instance of the **System.BadImageFormatException** class
2 with a specified error message and a reference to the inner exception that is the
3 root cause of this exception.

4 When an **Exception** *X* is thrown as a direct result of a previous exception *Y*,
5 the **System.Exception.InnerException** property of *X* should contain a reference
6 to *Y*. The **InnerException** property returns the same value as was passed into the
7 constructor, or **null** if the inner exception value was not supplied to the
8 constructor. The error message that explains the reason for the exception. An
9 instance of **System.Exception** that is the cause of the current **Exception**. If *inner*
10 is non-null, then the current **Exception** is raised in a catch block handling *inner*.

11 **BadImageFormatException**

12 *Example Syntax:*

13 **ToString**

14
15 [C#] public BadImageFormatException(string message, string fileName);
16 [C++] public: BadImageFormatException(String* message, String* fileName);
17 [VB] Public Sub New(ByVal message As String, ByVal fileName As String)
18 [JScript] public function BadImageFormatException(message : String, fileName :
19 String);

20
21 *Description*

22 Initializes a new instance of the **System.BadImageFormatException** class
23 with serialized data.

This constructor is called during deserialization to reconstitute the exception object transmitted over a stream. For more information, see . A reference to the inner exception.

BadImageFormatException

Example Syntax:

ToString

```
[C#] public BadImageFormatException(string message, string fileName,
Exception inner);
[C++] public: BadImageFormatException(String* message, String* fileName,
Exception* inner);
[VB] Public Sub New(ByVal message As String, ByVal fileName As String,
ByVal inner As Exception)
[JScript] public function BadImageFormatException(message : String, fileName :
String, inner : Exception);
```

Description

Initializes a new instance of the **System.BadImageFormatException** class with a specified error message and a reference to the inner exception that is the root cause of this exception.

When an **Exception** *X* is thrown as a direct result of a previous exception *Y*, the **System.Exception.InnerException** property of *X* should contain a reference to *Y*. The **InnerException** property returns the same value as was passed into the constructor, or **null** if the inner exception value was not supplied to the constructor. The error message that explains the reason for the exception. An

instance of **System.Exception** that is the cause of the current **Exception**. If *inner* is non-null, then the current **Exception** is raised in a catch block handling *inner* .

FileName

ToString

```
[C#] public string FileName {get;}
```

```
[C++] public: __property String* get_FileName();
```

```
[VB] Public ReadOnly Property FileName As String
```

```
[JScript] public function get FileName() : String;
```

Description

Gets the name of the file that causes this exception.

FusionLog

ToString

```
[C#] public string FusionLog {get;}
```

```
[C++] public: __property String* get_FusionLog();
```

```
[VB] Public ReadOnly Property FusionLog As String
```

```
[JScript] public function get FusionLog() : String;
```

Description

Gets the log file that describes why loading of an assembly failed.

HelpLink

HResult

InnerException

Message

ToString

Description

Gets the error message and the name of the file that caused this exception.

Source

StackTrace

TargetSite

GetObjectData

[C#] public override void GetObjectData(SerializationInfo info, StreamingContext context);

[C++] public: void GetObjectData(SerializationInfo* info, StreamingContext context);

[VB] Overrides Public Sub GetObjectData(ByVal info As SerializationInfo, ByVal context As StreamingContext)

[JScript] public override function GetObjectData(info : SerializationInfo, context : StreamingContext);

Description

Sets the **System.Runtime.Serialization.SerializationInfo** object with the file name, fusion log, and additional exception information.

ToString

```

1  [C#] public override string ToString();
2
3  [C++] public: String* ToString();
4
5  [VB] Overrides Public Function ToString() As String
6
7  [JScript] public override function ToString() : String;
8

```

Description

Returns the fully qualified name of this exception and possibly the error message, the name of the inner exception, and the stack trace.

Return Value: A string containing the fully qualified name of this exception and possibly the error message, the name of the inner exception, and the stack trace.

BitConverter class (System)

ToString

Description

Converts base data types to an array of bytes, and an array of bytes to base data types.

This class facilitates manipulating value types in their fundamental form. A byte is defined as an 8-bit unsigned integer.

ToString

```

23 [C#] public static readonly bool IsLittleEndian;
24 [C++] public: static bool IsLittleEndian;
25 [VB] Public Shared ReadOnly IsLittleEndian As Boolean

```

1 [JScript] public static var IsLittleEndian : Boolean;

3 *Description*

4 Indicates the byte order ("endianess") in which data is stored in this
5 computer architecture.

6 This value is **true** if the architecture is little-endian; **false** if it is big-endian.

7 DoubleToInt64Bits

9 [C#] public static long DoubleToInt64Bits(double value);

10 [C++] public: static __int64 DoubleToInt64Bits(double value);

11 [VB] Public Shared Function DoubleToInt64Bits(ByVal value As Double) As

12 Long

13 [JScript] public static function DoubleToInt64Bits(value : double) : long;

15 *Description*

16 Converts the specified double-precision floating point number to a 64-bit
17 signed integer.

18 *Return Value:* A 64-bit signed integer whose value is equivalent to *value* . The
19 number to convert.

20 GetBytes

22 [C#] public static byte[] GetBytes(bool value);

23 [C++] public: static unsigned char GetBytes(bool value) __gc[];

24 [VB] Public Shared Function GetBytes(ByVal value As Boolean) As Byte()

25 [JScript] public static function GetBytes(value : Boolean) : Byte[]; Converts the

specified data to an array of bytes.

Description

Returns the specified Boolean value as an array of bytes.

Return Value: An array of bytes with length 1. A Boolean value.

GetBytes

[C#] public static byte[] GetBytes(char value);

[C++] public: static unsigned char GetBytes(__wchar_t value) __gc[];

[VB] Public Shared Function GetBytes(ByVal value As Char) As Byte()

[JScript] public static function GetBytes(value : Char) : Byte[];

Description

Returns the specified Unicode character value as an array of bytes.

Return Value: An array of bytes with length 2. A character to convert.

GetBytes

[C#] public static byte[] GetBytes(double value);

[C++] public: static unsigned char GetBytes(double value) __gc[];

[VB] Public Shared Function GetBytes(ByVal value As Double) As Byte()

[JScript] public static function GetBytes(value : double) : Byte[];

Description

1 Returns the specified double-precision floating point value as an array of
2 bytes.

3 *Return Value:* An array of bytes with length 8. The number to convert.

4 GetBytes

6 [C#] public static byte[] GetBytes(short value);

7 [C++] public: static unsigned char GetBytes(short value) __gc[];

8 [VB] Public Shared Function GetBytes(ByVal value As Short) As Byte()

9 [JScript] public static function GetBytes(value : Int16) : Byte[];

11 *Description*

12 Returns the specified 16-bit signed integer value as an array of bytes.

13 *Return Value:* An array of bytes with length 2. The number to convert.

14 GetBytes

16 [C#] public static byte[] GetBytes(int value);

17 [C++] public: static unsigned char GetBytes(int value) __gc[];

18 [VB] Public Shared Function GetBytes(ByVal value As Integer) As Byte()

19 [JScript] public static function GetBytes(value : int) : Byte[];

21 *Description*

22 Returns the specified 32-bit signed integer value as an array of bytes.

23 *Return Value:* An array of bytes with length 4. The number to convert.

24 GetBytes

```

1
2 [C#] public static byte[] GetBytes(long value);
3 [C++] public: static unsigned char GetBytes(__int64 value) __gc[];
4 [VB] Public Shared Function GetBytes(ByVal value As Long) As Byte()
5 [JScript] public static function GetBytes(value : long) : Byte[];
6

```

Description

Returns the specified 64-bit signed integer value as an array of bytes.

Return Value: An array of bytes with length 8. The number to convert.

GetBytes

```

11
12 [C#] public static byte[] GetBytes(float value);
13 [C++] public: static unsigned char GetBytes(float value) __gc[];
14 [VB] Public Shared Function GetBytes(ByVal value As Single) As Byte()
15 [JScript] public static function GetBytes(value : float) : Byte[];
16

```

Description

Returns the specified single-precision floating point value as an array of bytes.

Return Value: An array of bytes with length 4. The number to convert.

GetBytes

```

22
23 [C#] public static byte[] GetBytes(ushort value);
24 [C++] public: static unsigned char GetBytes(unsigned short value) __gc[];
25 [VB] Public Shared Function GetBytes(ByVal value As UInt16) As Byte()

```

1 [JScript] public static function GetBytes(value : UInt16) : Byte[];

3 *Description*

4 Returns the specified 16-bit unsigned integer value as an array of bytes.

5 *Return Value:* An array of bytes with length 2. The number to convert.

6 GetBytes

8 [C#] public static byte[] GetBytes(uint value);

9 [C++] public: static unsigned char GetBytes(unsigned int value) __gc[];

10 [VB] Public Shared Function GetBytes(ByVal value As UInt32) As Byte()

11 [JScript] public static function GetBytes(value : UInt32) : Byte[];

13 *Description*

14 Returns the specified 32-bit unsigned integer value as an array of bytes.

15 *Return Value:* An array of bytes with length 4. The number to convert.

16 GetBytes

18 [C#] public static byte[] GetBytes(ulong value);

19 [C++] public: static unsigned char GetBytes(unsigned __int64 value) __gc[];

20 [VB] Public Shared Function GetBytes(ByVal value As UInt64) As Byte()

21 [JScript] public static function GetBytes(value : UInt64) : Byte[];

23 *Description*

24 Returns the specified 64-bit unsigned integer value as an array of bytes.

25 *Return Value:* An array of bytes with length 8. The number to convert.

Int64BitsToDouble

[C#] public static double Int64BitsToDouble(long value);

[C++] public: static double Int64BitsToDouble(__int64 value);

[VB] Public Shared Function Int64BitsToDouble(ByVal value As Long) As Double

[JScript] public static function Int64BitsToDouble(value : long) : double;

Description

Converts the specified 64-bit signed integer to a double-precision floating point number.

Return Value: A double-precision floating point number whose value is equivalent to *value* . The number to convert.

ToBoolean

[C#] public static bool ToBoolean(byte[] value, int startIndex);

[C++] public: static bool ToBoolean(unsigned char value __gc[], int startIndex);

[VB] Public Shared Function ToBoolean(ByVal value() As Byte, ByVal startIndex As Integer) As Boolean

[JScript] public static function ToBoolean(value : Byte[], startIndex : int) : Boolean;

Description

Returns a Boolean value converted from one byte at a specified position in a byte array.

1 *Return Value:* **true** if the byte at *startIndex* in *value* is nonzero; otherwise, **false** .

2 An array of bytes. The starting position within *value*.

3 ToChar

4
5 [C#] public static char ToChar(byte[] value, int startIndex);

6 [C++] public: static __wchar_t ToChar(unsigned char value __gc[], int startIndex);

7 [VB] Public Shared Function ToChar(Byte value() As Byte, ByVal startIndex

8 As Integer) As Char

9 [JScript] public static function ToChar(value : Byte[], startIndex : int) : Char;

11 Description

12 Returns a Unicode character converted from two bytes at a specified
13 position in a byte array.

14 *Return Value:* A character formed by two bytes beginning at *startIndex* . An array.
15 The starting position within *value* .

16 ToDouble

17
18 [C#] public static double ToDouble(byte[] value, int startIndex);

19 [C++] public: static double ToDouble(unsigned char value __gc[], int startIndex);

20 [VB] Public Shared Function ToDouble(Byte value() As Byte, ByVal startIndex

21 As Integer) As Double

22 [JScript] public static function ToDouble(value : Byte[], startIndex : int) : double;

24 Description

1 Returns a double-precision floating point number converted from eight
2 bytes at a specified position in a byte array.

3 *Return Value:* A double precision floating point number formed by eight bytes
4 beginning at *startIndex* . An array of bytes. The starting position within *value*.

5 ToInt16

6
7 [C#] public static short ToInt16(byte[] value, int startIndex);

8 [C++] public: static short ToInt16(unsigned char value __gc[], int startIndex);

9 [VB] Public Shared Function ToInt16(ByVal value() As Byte, ByVal startIndex
10 As Integer) As Short

11 [JScript] public static function ToInt16(value : Byte[], startIndex : int) : Int16;

12 Description

13
14 Returns a 16-bit signed integer converted from two bytes at a specified
15 position in a byte array.

16 *Return Value:* A 16-bit signed integer formed by two bytes beginning at *startIndex*
17 . An array of bytes. The starting position within *value*.

18 ToInt32

19
20 [C#] public static int ToInt32(byte[] value, int startIndex);

21 [C++] public: static int ToInt32(unsigned char value __gc[], int startIndex);

22 [VB] Public Shared Function ToInt32(ByVal value() As Byte, ByVal startIndex
23 As Integer) As Integer

24 [JScript] public static function ToInt32(value : Byte[], startIndex : int) : int;

Description

Returns a 32-bit signed integer converted from four bytes at a specified position in a byte array.

Return Value: A 32-bit signed integer formed by four bytes beginning at *startIndex.value* is **null** . An array of bytes. The starting position within *value*.

ToInt64

```
[C#] public static long ToInt64(byte[] value, int startIndex);
```

```
[C++] public: static __int64 ToInt64(unsigned char value __gc[], int startIndex);
```

```
[VB] Public Shared Function ToInt64(ByVal value() As Byte, ByVal startIndex  
As Integer) As Long
```

```
[JScript] public static function ToInt64(value : Byte[], startIndex : int) : long;
```

Description

Returns a 64-bit signed integer converted from eight bytes at a specified position in a byte array.

Return Value: A 64-bit signed integer formed by eight bytes beginning at *startIndex* . An array of bytes. The starting position within *value*.

ToSingle

```
[C#] public static float ToSingle(byte[] value, int startIndex);
```

```
[C++] public: static float ToSingle(unsigned char value __gc[], int startIndex);
```

```
[VB] Public Shared Function ToSingle(ByVal value() As Byte, ByVal startIndex  
As Integer) As Single
```


1 [JScript] public static function ToSingle(value : Byte[], startIndex : int) : float;

3 *Description*

4 Returns a single-precision floating point number converted from four bytes
5 at a specified position in a byte array.

6 *Return Value:* A single-precision floating point number formed by four bytes
7 beginning at *startIndex* . An array of bytes. The starting position within *value*.

8 ToString

10 [C#] public static string ToString(byte[] value);

11 [C++] public: static String* ToString(unsigned char value __gc[]);

12 [VB] Public Shared Function ToString(ByVal value() As Byte) As String

13 [JScript] public static function ToString(value : Byte[]) : String;

15 *Description*

16 Returns a **String** converted from the elements of a byte array.

17 *Return Value:* A **System.String** of hexadecimal pairs separated by hyphens, where
18 each pair represents the corresponding element in *value*; for example, "7F-2C-
19 4A".

20 All the elements of *value* are converted. An array of bytes.

21 ToString

23 [C#] public static string ToString(byte[] value, int startIndex);

24 [C++] public: static String* ToString(unsigned char value __gc[], int startIndex);

25 [VB] Public Shared Function ToString(ByVal value() As Byte, ByVal startIndex

As Integer) As String

[JScript] public static function ToString(value : Byte[], startIndex : int) : String;

Description

Returns a **String** converted from the elements of a byte array starting at a specified array position.

Return Value: A **System.String** of hexadecimal pairs separated by hyphens, where each pair represents the corresponding element in *value* ; for example, "7F-2C-4A".

The elements from array position *startIndex* to the end of the array are converted. An array of bytes. The starting position within *value*.

ToString

[C#] public static string ToString(byte[] value, int startIndex, int length);

[C++] public: static String* ToString(unsigned char value __gc[], int startIndex, int length);

[VB] Public Shared Function ToString(ByVal value() As Byte, ByVal startIndex As Integer, ByVal length As Integer) As String

[JScript] public static function ToString(value : Byte[], startIndex : int, length : int) : String; Returns a **String** converted from the elements of a byte array.

Description

Returns a **String** converted from a specified number of bytes at a specified position in a byte array.

Return Value: A **System.String** of hexadecimal pairs separated by hyphens, where

each pair represents the corresponding element in *value* ; for example, "7F-2C-4A".

The *length* elements from array position *startIndex* are converted. An array of bytes. The starting position within *value*. The number of bytes to convert.

ToUInt16

[C#] public static ushort ToUInt16(byte[] value, int startIndex);

[C++] public: static unsigned short ToUInt16(unsigned char value __gc[], int startIndex);

[VB] Public Shared Function ToUInt16(ByVal value() As Byte, ByVal startIndex As Integer) As UInt16

[JScript] public static function ToUInt16(value : Byte[], startIndex : int) : UInt16;

Description

Returns a 16-bit unsigned integer converted from two bytes at a specified position in a byte array.

Return Value: A 16-bit unsigned integer formed by two bytes beginning at *startIndex* . The array of bytes. The starting position within *value*.

ToUInt32

[C#] public static uint ToUInt32(byte[] value, int startIndex);

[C++] public: static unsigned int ToUInt32(unsigned char value __gc[], int startIndex);

[VB] Public Shared Function ToUInt32(ByVal value() As Byte, ByVal startIndex As Integer) As UInt32

1 [JScript] public static function ToUInt32(value : Byte[], startIndex : int) : UInt32;

3 *Description*

4 Returns a 32-bit unsigned integer converted from four bytes at a specified
5 position in a byte array.

6 *Return Value:* A 32-bit unsigned integer formed by four bytes beginning at
7 *startIndex* . An array of bytes. The starting position within *value*.

8 ToUInt64

10 [C#] public static ulong ToUInt64(byte[] value, int startIndex);

11 [C++] public: static unsigned __int64 ToUInt64(unsigned char value __gc[], int
12 startIndex);

13 [VB] Public Shared Function ToUInt64(Byte[] value) As Byte, ByVal startIndex
14 As Integer) As UInt64

15 [JScript] public static function ToUInt64(value : Byte[], startIndex : int) : UInt64;

17 *Description*

18 Returns a 64-bit unsigned integer converted from eight bytes at a specified
19 position in a byte array.

20 *Return Value:* A 64-bit unsigned integer formed by the eight bytes beginning at
21 *startIndex* . An array of bytes. The starting position within *value*.

22 Boolean structure (System)

23 ToUInt64

Description

Represents a boolean value.

Instances of this type have values of either **true** or **false** .

ToUInt64

[C#] public static readonly string FalseString;

[C++] public: static String* FalseString;

[VB] Public Shared ReadOnly FalseString As String

[JScript] public static var FalseString : String;

Description

Represents the boolean value **false** as a **System.String** . This field is read-only.

This field is equal to the **System.String** "False".

ToUInt64

[C#] public static readonly string TrueString;

[C++] public: static String* TrueString;

[VB] Public Shared ReadOnly TrueString As String

[JScript] public static var TrueString : String;

Description

Represents the boolean value **true** as a **System.String** . This field is read-only.

This field is equal to the **System.String** "True".

CompareTo

[C#] public int CompareTo(object obj);

[C++] public: __sealed int CompareTo(Object* obj);

[VB] NotOverridable Public Function CompareTo(ByVal obj As Object) As

Integer

[JScript] public function CompareTo(obj : Object) : int;

Description

Compares this instance to a specified object and returns an indication of their relative values.

Return Value: A signed integer that indicates the relative order of this instance and *obj* .

obj must be **null** or an instance of **Boolean** ; otherwise, an exception is thrown. An **System.Object** to compare to this instance. It may be a null reference.

Equals

[C#] public override bool Equals(object obj);

[C++] public: bool Equals(Object* obj);

[VB] Overrides Public Function Equals(ByVal obj As Object) As Boolean

[JScript] public override function Equals(obj : Object) : Boolean;

Description

Returns a value indicating whether this instance is equal to a specified object.

Return Value: **true** if *obj* is a **Boolean** and has the same value as this instance; otherwise, **false** .

This method overrides **System.Object.Equals(System.Object)** . An **System.Object** to compare to this instance.

GetHashCode

[C#] public override int GetHashCode();

[C++] public: int GetHashCode();

[VB] Overrides Public Function GetHashCode() As Integer

[JScript] public override function GetHashCode() : int;

Description

Returns the hash code for this instance.

Return Value: A hash code for the current **System.Boolean** .

The **System.Boolean** class implements **true** as the integer, one, and **false** as the integer, zero. However, a particular programming language might represent **true** and **false** with other values.

GetTypeCode

[C#] public TypeCode GetTypeCode();

[C++] public: __sealed TypeCode GetTypeCode();

1 [VB] NotOverridable Public Function GetTypeCode() As TypeCode

2 [JScript] public function GetTypeCode() : TypeCode;

3
4 *Description*

5 Returns the **TypeCode** for value type **Boolean** .

6 *Return Value:* The enumerated constant, **System.TypeCode.Boolean** .

7 **Parse**

8
9 [C#] public static bool Parse(string value);

10 [C++] public: static bool Parse(String* value);

11 [VB] Public Shared Function Parse(ByVal value As String) As Boolean

12 [JScript] public static function Parse(value : String) : Boolean;

13
14 *Description*

15 Converts the specified **System.String** representation of a logical value to
16 its **System.Boolean** equivalent.

17 *Return Value:* **true** if *value* is equivalent to **System.Boolean.TrueString** ;
18 otherwise **false**. *value* is a null reference.

19 The *value* parameter , optionally preceded or trailed by white space, must
20 contain either **TrueString** or **FalseString** ; otherwise, an exception is thrown. The
21 comparison is case-insensitive. A **System.String** containing the value to convert.

22 **IConvertible.ToBoolean**

23
24 [C#] bool IConvertible.ToBoolean(IFormatProvider provider);

25 [C++] bool IConvertible::ToBoolean(IFormatProvider* provider);


```

1  [VB] Function ToBoolean(ByVal provider As IFormatProvider) As Boolean
2  Implements IConvertible.ToBoolean
3  [JScript] function IConvertible.ToBoolean(provider : IFormatProvider) : Boolean;
4      IConvertible.ToByte
5
6  [C#] byte IConvertible.ToByte(IFormatProvider provider);
7  [C++] unsigned char IConvertible::ToByte(IFormatProvider* provider);
8  [VB] Function ToByte(ByVal provider As IFormatProvider) As Byte Implements
9  IConvertible.ToByte
10 [JScript] function IConvertible.ToByte(provider : IFormatProvider) : Byte;
11     IConvertible.ToChar
12
13 [C#] char IConvertible.ToChar(IFormatProvider provider);
14 [C++] __wchar_t IConvertible::ToChar(IFormatProvider* provider);
15 [VB] Function ToChar(ByVal provider As IFormatProvider) As Char Implements
16 IConvertible.ToChar
17 [JScript] function IConvertible.ToChar(provider : IFormatProvider) : Char;
18     IConvertible.ToDateTime
19
20 [C#] DateTime IConvertible.ToDateTime(IFormatProvider provider);
21 [C++] DateTime IConvertible::ToDateTime(IFormatProvider* provider);
22 [VB] Function ToDateTime(ByVal provider As IFormatProvider) As DateTime
23 Implements IConvertible.ToDateTime
24 [JScript] function IConvertible.ToDateTime(provider : IFormatProvider) :
25 DateTime;
```

1 IConvertible.ToDecimal

3 [C#] decimal IConvertible.ToDecimal(IFORMATPROVIDER provider);

4 [C++] Decimal IConvertible::ToDecimal(IFORMATPROVIDER* provider);

5 [VB] Function ToDecimal(ByVal provider As IFORMATPROVIDER) As Decimal

6 Implements IConvertible.ToDecimal

7 [JScript] function IConvertible.ToDecimal(provider : IFORMATPROVIDER) : Decimal;

8 IConvertible.ToDouble

10 [C#] double IConvertible.ToDouble(IFORMATPROVIDER provider);

11 [C++] double IConvertible::ToDouble(IFORMATPROVIDER* provider);

12 [VB] Function ToDouble(ByVal provider As IFORMATPROVIDER) As Double

13 Implements IConvertible.ToDouble

14 [JScript] function IConvertible.ToDouble(provider : IFORMATPROVIDER) : double;

15 IConvertible.ToInt16

17 [C#] short IConvertible.ToInt16(IFORMATPROVIDER provider);

18 [C++] short IConvertible::ToInt16(IFORMATPROVIDER* provider);

19 [VB] Function ToInt16(ByVal provider As IFORMATPROVIDER) As Short

20 Implements IConvertible.ToInt16

21 [JScript] function IConvertible.ToInt16(provider : IFORMATPROVIDER) : Int16;

22 IConvertible.ToInt32

24 [C#] int IConvertible.ToInt32(IFORMATPROVIDER provider);

25 [C++] int IConvertible::ToInt32(IFORMATPROVIDER* provider);

```

1 [VB] Function ToInt32(ByVal provider As IFormatProvider) As Integer
2 Implements IConvertible.ToInt32
3 [JScript] function IConvertible.ToInt32(provider : IFormatProvider) : int;
4     IConvertible.ToInt64
5
6 [C#] long IConvertible.ToInt64(IFormatProvider provider);
7 [C++] __int64 IConvertible::ToInt64(IFormatProvider* provider);
8 [VB] Function ToInt64(ByVal provider As IFormatProvider) As Long Implements
9 IConvertible.ToInt64
10 [JScript] function IConvertible.ToInt64(provider : IFormatProvider) : long;
11     IConvertible.ToSByte
12
13 [C#] sbyte IConvertible.ToSByte(IFormatProvider provider);
14 [C++] char IConvertible::ToSByte(IFormatProvider* provider);
15 [VB] Function ToSByte(ByVal provider As IFormatProvider) As SByte
16 Implements IConvertible.ToSByte
17 [JScript] function IConvertible.ToSByte(provider : IFormatProvider) : SByte;
18     IConvertible.ToSingle
19
20 [C#] float IConvertible.ToSingle(IFormatProvider provider);
21 [C++] float IConvertible::ToSingle(IFormatProvider* provider);
22 [VB] Function ToSingle(ByVal provider As IFormatProvider) As Single
23 Implements IConvertible.ToSingle
24 [JScript] function IConvertible.ToSingle(provider : IFormatProvider) : float;
25     IConvertible.ToType

```

```

1
2 [C#] object IConvertible.ToType(Type type, IFormatProvider provider);
3 [C++] Object* IConvertible::ToType(Type* type, IFormatProvider* provider);
4 [VB] Function ToType(ByVal type As Type, ByVal provider As IFormatProvider)
5 As Object Implements IConvertible.ToType
6 [JScript] function IConvertible.ToType(type : Type, provider : IFormatProvider) :
7 Object;
8     IConvertible.ToUInt16
9
10 [C#] ushort IConvertible.ToUInt16(IFormatProvider provider);
11 [C++] unsigned short IConvertible::ToUInt16(IFormatProvider* provider);
12 [VB] Function ToUInt16(ByVal provider As IFormatProvider) As UInt16
13 Implements IConvertible.ToUInt16
14 [JScript] function IConvertible.ToUInt16(provider : IFormatProvider) : UInt16;
15     IConvertible.ToUInt32
16
17 [C#] uint IConvertible.ToUInt32(IFormatProvider provider);
18 [C++] unsigned int IConvertible::ToUInt32(IFormatProvider* provider);
19 [VB] Function ToUInt32(ByVal provider As IFormatProvider) As UInt32
20 Implements IConvertible.ToUInt32
21 [JScript] function IConvertible.ToUInt32(provider : IFormatProvider) : UInt32;
22     IConvertible.ToUInt64
23
24 [C#] ulong IConvertible.ToUInt64(IFormatProvider provider);
25 [C++] unsigned __int64 IConvertible::ToUInt64(IFormatProvider* provider);

```

1 [VB] Function ToUInt64(ByVal provider As IFormatProvider) As UInt64

2 Implements IConvertible.ToUInt64

3 [JScript] function IConvertible.ToUInt64(provider : IFormatProvider) : UInt64;

4 ToString

6 [C#] public override string ToString();

7 [C++] public: String* ToString();

8 [VB] Overrides Public Function ToString() As String

9 [JScript] public override function ToString() : String; Converts the value of this
10 instance to its equivalent **System.String** representation.

12 *Description*

13 Converts the value of this instance to its equivalent **System.String**
14 representation.

15 *Return Value:* **System.Boolean.TrueString** if the value of this instance is **true** ,
16 or **System.Boolean.FalseString** if the value of this instance is **false** .

17 ToString

19 [C#] public string ToString(IFormatProvider provider);

20 [C++] public: __sealed String* ToString(IFormatProvider* provider);

21 [VB] NotOverridable Public Function ToString(ByVal provider As
22 IFormatProvider) As String

23 [JScript] public function ToString(provider : IFormatProvider) : String;

25 *Description*

Converts the value of this instance to its equivalent **String** representation.

Return Value: **System.Boolean.TrueString** if the value of this instance is **true** ,
or **System.Boolean.FalseString** if the value of this instance is **false** .

The *provider* parameter is reserved. It does not participate in the execution
of this method. (Reserved) An **System.IFormatProvider** object.

Buffer class (System)

ToString

Description

Manipulates unmanaged memory represented as arrays of bytes.

This class provides methods to copy bytes from one primitive array to
another primitive array without respecting types, get a byte from an array, set a
byte in an array, and obtain the length of an array.

BlockCopy

[C#] public static void BlockCopy(Array src, int srcOffset, Array dst, int
dstOffset, int count);

[C++] public: static void BlockCopy(Array* src, int srcOffset, Array* dst, int
dstOffset, int count);

[VB] Public Shared Sub BlockCopy(ByVal src As Array, ByVal srcOffset As
Integer, ByVal dst As Array, ByVal dstOffset As Integer, ByVal count As Integer)

[JScript] public static function BlockCopy(src : Array, srcOffset : int, dst : Array,
dstOffset : int, count : int);

Description

Copies a specified number of bytes from a source array starting at a particular offset to a destination array starting at a particular offset.

Copies *count* bytes from *src*, beginning at *srcOffset*, to *dst*, beginning at *dstOffset*. The source buffer. The byte offset into *src*. The destination buffer. The byte offset into *dst*. The number of bytes to copy.

ByteLength

[C#] public static int ByteLength(Array array);

[C++] public: static int ByteLength(Array* array);

[VB] Public Shared Function ByteLength(ByVal array As Array) As Integer

[JScript] public static function ByteLength(array : Array) : int;

Description

Returns the number of bytes in the specified array.

Return Value: The number of bytes in the array. An array.

GetByte

[C#] public static byte GetByte(Array array, int index);

[C++] public: static unsigned char GetByte(Array* array, int index);

[VB] Public Shared Function GetByte(ByVal array As Array, ByVal index As Integer) As Byte

[JScript] public static function GetByte(array : Array, index : int) : Byte;

Description

Retrieves the byte at a specified location in a specified array.

Return Value: Returns the *index* byte in the array.

The **GetByte** method gets a particular byte out of the array. The array must be an array of primitives. An array. A location in the array.

SetByte

[C#] public static void SetByte(Array array, int index, byte value);

[C++] public: static void SetByte(Array* array, int index, unsigned char value);

[VB] Public Shared Sub SetByte(ByVal array As Array, ByVal index As Integer, ByVal value As Byte)

[JScript] public static function SetByte(array : Array, index : int, value : Byte);

Description

Assigns a specified value to a byte at a particular location in a specified array.

array must be an array of primitives. An array. A location in the array. A value to assign.

Byte structure (System)

ToString

Description

Represents an 8-bit unsigned integer.

The **Byte** value type represents unsigned integers with values ranging from 0 to 255.

ToString

[C#] public const byte MaxValue;

[C++] public: const unsigned char MaxValue;

[VB] Public Const MaxValue As Byte

[JScript] public var MaxValue : Byte;

Description

A constant representing the largest possible value of **Byte** .

The value of this constant is 255; that is, hexadecimal 0xFF.

ToString

[C#] public const byte MinValue;

[C++] public: const unsigned char MinValue;

[VB] Public Const MinValue As Byte

[JScript] public var MinValue : Byte;

Description

A constant representing the smallest possible value of **Byte** .

The value of this constant is 0.

CompareTo

[C#] public int CompareTo(object value);

1 [C++] public: __sealed int CompareTo(Object* value);

2 [VB] NotOverridable Public Function CompareTo(ByVal value As Object) As

3 Integer

4 [JScript] public function CompareTo(value : Object) : int;

6 *Description*

7 Compares this instance to a specified object and returns an indication of
8 their relative values.

9 *Return Value:* A signed number indicating the relative values of this instance and
10 *value* .

11 Any instance of **Byte** , regardless of its value, is considered greater than
12 **null** . An object to compare, or **null**.

13 Equals

15 [C#] public override bool Equals(object obj);

16 [C++] public: bool Equals(Object* obj);

17 [VB] Overrides Public Function Equals(ByVal obj As Object) As Boolean

18 [JScript] public override function Equals(obj : Object) : Boolean;

20 *Description*

21 Returns a value indicating whether this instance is equal to a specified
22 object.

23 *Return Value:* **true** if *obj* is an instance of **Byte** and equals the value of this
24 instance; otherwise, **false** . An object to compare with this instance or **null**.

25 GetHashCode

1
2 [C#] public override int GetHashCode();

3 [C++] public: int GetHashCode();

4 [VB] Overrides Public Function GetHashCode() As Integer

5 [JScript] public override function GetHashCode() : int;

6
7 *Description*

8 Returns the hash code for this instance.

9 *Return Value:* A 32-bit signed integer hash code.

10 **GetTypeCode**

11
12 [C#] public TypeCode GetTypeCode();

13 [C++] public: __sealed TypeCode GetTypeCode();

14 [VB] NotOverridable Public Function GetTypeCode() As TypeCode

15 [JScript] public function GetTypeCode() : TypeCode;

16
17 *Description*

18 Returns the **TypeCode** for value type **Byte** .

19 *Return Value:* The enumerated constant, **System.TypeCode.Byte** .

20 **Parse**

21
22 [C#] public static byte Parse(string s);

23 [C++] public: static unsigned char Parse(String* s);

24 [VB] Public Shared Function Parse(ByVal s As String) As Byte

25 [JScript] public static function Parse(s : String) : Byte; Converts the **String**

representation of a number to its 8-bit unsigned integer equivalent.

Description

Converts the **String** representation of a number to its 8-bit unsigned integer equivalent.

Return Value: An 8-bit unsigned integer equivalent to the number contained in *s*.

s contains a number of the form: [ws][sign]digits[ws] Items in square brackets ('[' and ']') are optional, and other items are as follows. A **System.String** containing a number to convert.

Parse

[C#] public static byte Parse(string s, IFormatProvider provider);

[C++] public: static unsigned char Parse(String* s, IFormatProvider* provider);

[VB] Public Shared Function Parse(ByVal s As String, ByVal provider As IFormatProvider) As Byte

[JScript] public static function Parse(s : String, provider : IFormatProvider) : Byte;

Description

Converts the **String** representation of a number in a specified culture-specific format to its 8-bit unsigned integer equivalent.

Return Value: An 8-bit unsigned integer equivalent to the number specified in *s*.

s contains a number of the form: [ws][sign]digits[ws] Items in square brackets ('[' and ']') are optional, and other items are as follows. A **System.String** containing a number to convert. An **System.IFormatProvider** interface implementation which supplies culture-specific formatting information about *s*.

Parse

```
[C#] public static byte Parse(string s, NumberStyles style);  
[C++] public: static unsigned char Parse(String* s, NumberStyles style);  
[VB] Public Shared Function Parse(ByVal s As String, ByVal style As  
NumberStyles) As Byte  
[JScript] public static function Parse(s : String, style : NumberStyles) : Byte;
```

Description

Converts the **String** representation of a number in a specified style to its 8-bit unsigned integer equivalent.

Return Value: An 8-bit unsigned integer equivalent to the number specified in *s*.

s contains a number of the form: [ws][sign]digits[ws] Items in square brackets ('[' and ']') are optional, and other items are as follows. A **System.String** containing a number to convert. The combination of one or more **System.Globalization.NumberStyles** constants that indicate the permitted format of *s*.

Parse

```
[C#] public static byte Parse(string s, NumberStyles style, IFormatProvider  
provider);  
[C++] public: static unsigned char Parse(String* s, NumberStyles style,  
IFormatProvider* provider);  
[VB] Public Shared Function Parse(ByVal s As String, ByVal style As  
NumberStyles, ByVal provider As IFormatProvider) As Byte
```

1 [JScript] public static function Parse(s : String, style : NumberStyles, provider :
2 IFormatProvider) : Byte;

4 *Description*

5 Converts the **String** representation of a number in a specified style and
6 culture-specific format to its 8-bit unsigned integer equivalent.

7 *Return Value:* An 8-bit unsigned integer equivalent to the number specified in *s* .

8 *s* contains a number of the form: [ws][sign]digits[ws] Items in square
9 brackets ('[' and ']') are optional, and other items are as follows. A **System.String**
10 containing a number to convert. The combination of one or more
11 **System.Globalization.NumberStyles** constants that indicate the permitted format
12 of *s*. An **System.IFormatProvider** interface implementation which supplies
13 culture-specific formatting information about *s*.

14 **IConvertible.ToBoolean**

15
16 [C#] bool IConvertible.ToBoolean(IFormatProvider provider);

17 [C++] bool IConvertible::ToBoolean(IFormatProvider* provider);

18 [VB] Function ToBoolean(ByVal provider As IFormatProvider) As Boolean

19 Implements IConvertible.ToBoolean

20 [JScript] function IConvertible.ToBoolean(provider : IFormatProvider) : Boolean;

21 **IConvertible.ToByte**

22
23 [C#] byte IConvertible.ToByte(IFormatProvider provider);

24 [C++] unsigned char IConvertible::ToByte(IFormatProvider* provider);

25 [VB] Function ToByte(ByVal provider As IFormatProvider) As Byte Implements

1 IConvertible.ToByte

2 [JScript] function IConvertible.ToByte(provider : IFormatProvider) : Byte;

3 IConvertible.ToChar

4
5 [C#] char IConvertible.ToChar(IFormatProvider provider);

6 [C++] __wchar_t IConvertible::ToChar(IFormatProvider* provider);

7 [VB] Function ToChar(ByVal provider As IFormatProvider) As Char Implements

8 IConvertible.ToChar

9 [JScript] function IConvertible.ToChar(provider : IFormatProvider) : Char;

10 IConvertible.ToDateTime

11
12 [C#] DateTime IConvertible.ToDateTime(IFormatProvider provider);

13 [C++] DateTime IConvertible::ToDateTime(IFormatProvider* provider);

14 [VB] Function ToDateTime(ByVal provider As IFormatProvider) As DateTime

15 Implements IConvertible.ToDateTime

16 [JScript] function IConvertible.ToDateTime(provider : IFormatProvider) :

17 DateTime;

18 IConvertible.ToDecimal

19
20 [C#] decimal IConvertible.ToDecimal(IFormatProvider provider);

21 [C++] Decimal IConvertible::ToDecimal(IFormatProvider* provider);

22 [VB] Function ToDecimal(ByVal provider As IFormatProvider) As Decimal

23 Implements IConvertible.ToDecimal

24 [JScript] function IConvertible.ToDecimal(provider : IFormatProvider) : Decimal;

25 IConvertible.ToDouble

```

1
2 [C#] double IConvertible.ToDouble(IFormatProvider provider);
3 [C++] double IConvertible::ToDouble(IFormatProvider* provider);
4 [VB] Function ToDouble(ByVal provider As IFormatProvider) As Double
5 Implements IConvertible.ToDouble
6 [JScript] function IConvertible.ToDouble(provider : IFormatProvider) : double;
7     IConvertible.ToInt16
8
9 [C#] short IConvertible.ToInt16(IFormatProvider provider);
10 [C++] short IConvertible::ToInt16(IFormatProvider* provider);
11 [VB] Function ToInt16(ByVal provider As IFormatProvider) As Short
12 Implements IConvertible.ToInt16
13 [JScript] function IConvertible.ToInt16(provider : IFormatProvider) : Int16;
14     IConvertible.ToInt32
15
16 [C#] int IConvertible.ToInt32(IFormatProvider provider);
17 [C++] int IConvertible::ToInt32(IFormatProvider* provider);
18 [VB] Function ToInt32(ByVal provider As IFormatProvider) As Integer
19 Implements IConvertible.ToInt32
20 [JScript] function IConvertible.ToInt32(provider : IFormatProvider) : int;
21     IConvertible.ToInt64
22
23 [C#] long IConvertible.ToInt64(IFormatProvider provider);
24 [C++] __int64 IConvertible::ToInt64(IFormatProvider* provider);
25 [VB] Function ToInt64(ByVal provider As IFormatProvider) As Long Implements

```


1 IConvertible.ToInt64

2 [JScript] function IConvertible.ToInt64(provider : IFormatProvider) : long;

3 IConvertible.ToSByte

5 [C#] sbyte IConvertible.ToSByte(IFormatProvider provider);

6 [C++] char IConvertible::ToSByte(IFormatProvider* provider);

7 [VB] Function ToSByte(ByVal provider As IFormatProvider) As SByte

8 Implements IConvertible.ToSByte

9 [JScript] function IConvertible.ToSByte(provider : IFormatProvider) : SByte;

10 IConvertible.ToSingle

12 [C#] float IConvertible.ToSingle(IFormatProvider provider);

13 [C++] float IConvertible::ToSingle(IFormatProvider* provider);

14 [VB] Function ToSingle(ByVal provider As IFormatProvider) As Single

15 Implements IConvertible.ToSingle

16 [JScript] function IConvertible.ToSingle(provider : IFormatProvider) : float;

17 IConvertible.ToType

19 [C#] object IConvertible.ToType(Type type, IFormatProvider provider);

20 [C++] Object* IConvertible::ToType(Type* type, IFormatProvider* provider);

21 [VB] Function ToType(ByVal type As Type, ByVal provider As IFormatProvider)

22 As Object Implements IConvertible.ToType

23 [JScript] function IConvertible.ToType(type : Type, provider : IFormatProvider) :

24 Object;

25 IConvertible.ToUInt16

```

1
2 [C#] ushort IConvertible.ToUInt16(IFormatProvider provider);
3 [C++] unsigned short IConvertible::ToUInt16(IFormatProvider* provider);
4 [VB] Function ToUInt16(ByVal provider As IFormatProvider) As UInt16
5 Implements IConvertible.ToUInt16
6 [JScript] function IConvertible.ToUInt16(provider : IFormatProvider) : UInt16;
7     IConvertible.ToUInt32
8
9 [C#] uint IConvertible.ToUInt32(IFormatProvider provider);
10 [C++] unsigned int IConvertible::ToUInt32(IFormatProvider* provider);
11 [VB] Function ToUInt32(ByVal provider As IFormatProvider) As UInt32
12 Implements IConvertible.ToUInt32
13 [JScript] function IConvertible.ToUInt32(provider : IFormatProvider) : UInt32;
14     IConvertible.ToUInt64
15
16 [C#] ulong IConvertible.ToUInt64(IFormatProvider provider);
17 [C++] unsigned __int64 IConvertible::ToUInt64(IFormatProvider* provider);
18 [VB] Function ToUInt64(ByVal provider As IFormatProvider) As UInt64
19 Implements IConvertible.ToUInt64
20 [JScript] function IConvertible.ToUInt64(provider : IFormatProvider) : UInt64;
21     ToString
22
23 [C#] public override string ToString();
24 [C++] public: String* ToString();
25 [VB] Overrides Public Function ToString() As String

```

1 [JScript] public override function ToString() : String; Converts the numeric value
2 of this instance to its equivalent **String** representation.

3
4 *Description*

5 Converts the numeric value of this instance to its equivalent **String**
6 representation.

7 *Return Value:* The **System.String** representation of the value of this instance,
8 consisting of a sequence of digits ranging from 0 to 9, without a sign or leading
9 zeroes.

10 The return value is formatted with the general format specifier ("G") and
11 the **System.Globalization.NumberFormatInfo** for the current culture.

12 **ToString**

13
14 [C#] public string ToString(IFormatProvider provider);

15 [C++] public: __sealed String* ToString(IFormatProvider* provider);

16 [VB] NotOverridable Public Function ToString(ByVal provider As
17 IFormatProvider) As String

18 [JScript] public function ToString(provider : IFormatProvider) : String;

19
20 *Description*

21 Converts the numeric value of this instance to its equivalent **String**
22 representation using the specified culture-specific format information.

23 *Return Value:* The **System.String** representation of the value of this instance as
24 specified by *provider* .

This instance is formatted with the general format specifier ("G"). An **System.IFormatProvider** interface implementation which supplies culture-specific formatting information.

ToString

```
[C#] public string ToString(string format);  
[C++] public: String* ToString(String* format);  
[VB] Public Function ToString(ByVal format As String) As String  
[JScript] public function ToString(format : String) : String;
```

Description

Converts the numeric value of this instance to its equivalent **String** representation using the specified format.

Return Value: The **System.String** representation of the value of this instance as specified by *format* .

If *format* is **null** or an empty string, the return value of this instance is formatted with the general format specifier ("G"). A format string.

ToString

```
[C#] public string ToString(string format, IFormatProvider provider);  
[C++] public: __sealed String* ToString(String* format, IFormatProvider*  
provider);  
[VB] NotOverridable Public Function ToString(ByVal format As String, ByVal  
provider As IFormatProvider) As String  
[JScript] public function ToString(format : String, provider : IFormatProvider) :
```

String;

Description

Converts the numeric value of this instance to its equivalent **String** representation using the specified format and culture-specific format information.

Return Value: The **System.String** representation of the value of this instance as specified by *format* and *provider* .

If *format* is **null** or an empty string, the return value for this instance is formatted with the general format specifier ("G"). A format specification. An **System.IFormatProvider** interface implementation which supplies culture-specific formatting information.

CannotUnloadAppDomainException class (System)

ToString

Description

The exception that is thrown when an attempt to unload an application domain fails.

System.CannotUnloadAppDomainException is thrown when there is an attempt to unload: The default application domain, which must remains loaded during the lifetime of the application.

CannotUnloadAppDomainException

Example Syntax:

ToString

```

1  [C#] public CannotUnloadAppDomainException();
2
3  [C++] public: CannotUnloadAppDomainException();
4
5  [VB] Public Sub New()
6
7  [JScript] public function CannotUnloadAppDomainException(); Initializes a new
8  instance of the System.CannotUnloadAppDomainException class.
9

```

Description

Initializes a new instance of the **System.CannotUnloadAppDomainException** class with default properties.

The following table shows the initial property values for an instance of **System.CannotUnloadAppDomainException**.

CannotUnloadAppDomainException

Example Syntax:

ToString

```

17 [C#] public CannotUnloadAppDomainException(string message);
18
19 [C++] public: CannotUnloadAppDomainException(String* message);
20
21 [VB] Public Sub New(ByVal message As String)
22
23 [JScript] public function CannotUnloadAppDomainException(message : String);
24

```

Description

Initializes a new instance of the **System.CannotUnloadAppDomainException** class with a specified error message.

The following table shows the initial property values for an instance of **System.CannotUnloadAppDomainException** . The error message that explains the reason for the exception.

CannotUnloadAppDomainException

Example Syntax:

ToString

[C#] protected CannotUnloadAppDomainException(SerializationInfo info, StreamingContext context);

[C++] protected: CannotUnloadAppDomainException(SerializationInfo* info, StreamingContext context);

[VB] Protected Sub New(ByVal info As SerializationInfo, ByVal context As StreamingContext)

[JScript] protected function CannotUnloadAppDomainException(info : SerializationInfo, context : StreamingContext);

Description

Initializes a new instance of the **System.CannotUnloadAppDomainException** class from serialized data.

This constructor is called during deserialization to reconstitute the exception object transmitted over a stream. For more information, see . The object that holds the serialized object data. The contextual information about the source or destination.

CannotUnloadAppDomainException

Example Syntax:

ToString

```
[C#] public CannotUnloadAppDomainException(string message, Exception
innerException);
[C++] public: CannotUnloadAppDomainException(String* message, Exception*
innerException);
[VB] Public Sub New(ByVal message As String, ByVal innerException As
Exception)
[JavaScript] public function CannotUnloadAppDomainException(message : String,
innerException : Exception);
```

Description

Initializes a new instance of the **System.CannotUnloadAppDomainException** class with a specified error message and a reference to the inner exception that is the root cause of this exception.

When an **Exception** *X* is thrown as a direct result of a previous exception *Y*, the **System.Exception.InnerException** property of *X* should contain a reference to *Y*. The **InnerException** property returns the same value as was passed into the constructor, or **null** if the inner exception value was not supplied to the constructor. The error message that explains the reason for the exception. An instance of **System.Exception** that is the cause of the current **Exception**. If *innerException* is non-null, then the current **Exception** is raised in a catch block handling *innerException*.

HelpLink

1 HRESULT
 2 InnerException
 3 Message
 4 Source
 5 StackTrace
 6 TargetSite
 7 Char structure (System)
 8 ToString

9
 10
 11 *Description*

12 Represents a Unicode character.
 13 The **Char** value type represents Unicode characters with values ranging
 14 from hexadecimal 0x0000 to 0xFFFF.

15 ToString
 16
 17 [C#] public const char MaxValue;
 18 [C++] public: const __wchar_t MaxValue;
 19 [VB] Public Const MaxValue As Char
 20 [JScript] public var MaxValue : Char;

21
 22 *Description*

23 A constant representing the largest possible value of **Char** .
 24 The value of this constant is hexadecimal 0xFFFF.
 25 ToString

```

1 [C#] public const char MinValue;
2
3 [C++] public: const __wchar_t MinValue;
4
5 [VB] Public Const MinValue As Char
6
7 [JScript] public var MinValue : Char;
8

```

Description

A constant representing the smallest possible value of **Char** .

The value of this constant is hexadecimal 0x00.

CompareTo

```

11
12 [C#] public int CompareTo(object value);
13
14 [C++] public: __sealed int CompareTo(Object* value);
15
16 [VB] NotOverridable Public Function CompareTo(ByVal value As Object) As
17

```

Integer

```

18 [JScript] public function CompareTo(value : Object) : int;
19

```

Description

Compares this instance to a specified object and returns an indication of their relative values.

Return Value: A signed number indicating the relative values of this instance and *value* .

Any instance of **Char** , regardless of its value, is considered greater than **null** . An object to compare, or **null**.

Equals

1
2 [C#] public override bool Equals(object obj);

3 [C++] public: bool Equals(Object* obj);

4 [VB] Overrides Public Function Equals(ByVal obj As Object) As Boolean

5 [JScript] public override function Equals(obj : Object) : Boolean;

6
7 *Description*

8 Returns a value indicating whether this instance is equal to a specified
9 object.

10 *Return Value:* **true** if *obj* is an instance of **Char** and equals the value of this
11 instance; otherwise, **false** . An object to compare with this instance or **null**.

12 GetHashCode

13
14 [C#] public override int GetHashCode();

15 [C++] public: int GetHashCode();

16 [VB] Overrides Public Function GetHashCode() As Integer

17 [JScript] public override function GetHashCode() : int;

18
19 *Description*

20 Returns the hash code for this instance.

21 *Return Value:* A 32-bit signed integer hash code.

22 GetNumericValue

23
24 [C#] public static double GetNumericValue(char c);

25 [C++] public: static double GetNumericValue(__wchar_t c);

[VB] Public Shared Function GetNumericValue(ByVal c As Char) As Double
[JScript] public static function GetNumericValue(c : Char) : double; Converts a
specified numeric Unicode character to a double-precision floating point number.

Description

Converts the specified numeric Unicode character to a double-precision
floating point number.

Return Value: The numeric value of *c* if that character represents a number;
otherwise, -1.

For example, if *c* is '5', the return value is 5. However, if *c* is 'z', the return
value is -1. A Unicode character.

GetNumericValue

[C#] public static double GetNumericValue(string s, int index);

[C++] public: static double GetNumericValue(String* s, int index);

[VB] Public Shared Function GetNumericValue(ByVal s As String, ByVal index
As Integer) As Double

[JScript] public static function GetNumericValue(s : String, index : int) : double;

Description

Converts the numeric Unicode character at the specified position in a
specified **System.String** to a double-precision floating point number.

Return Value: The numeric value of the character at position *index* in *s* if that
character represents a number; otherwise, -1.

For example, if the character at position *index* in *s* is '5', the return value is 5. However, if the character at position *index* in *s* is 'z', the return value is -1. A **System.String**. A 32-bit signed integer that specifies a character position in *s*.

GetTypeCode

```
[C#] public TypeCode GetTypeCode();
[C++] public: __sealed TypeCode GetTypeCode();
[VB] NotOverridable Public Function GetTypeCode() As TypeCode
[JScript] public function GetTypeCode() : TypeCode;
```

Description

Returns the **TypeCode** for value type **Char** .

Return Value: The enumerated constant, **System.TypeCode.Char** .

GetUnicodeCategory

```
[C#] public static UnicodeCategory GetUnicodeCategory(char c);
[C++] public: static UnicodeCategory GetUnicodeCategory(__wchar_t c);
[VB] Public Shared Function GetUnicodeCategory(ByVal c As Char) As
UnicodeCategory
[JScript] public static function GetUnicodeCategory(c : Char) : UnicodeCategory;
```

Categorizes a Unicode character into a group identified by a **UnicodeCategory** enumerated constant.

Description

1 Categorizes a specified Unicode character into a group identified by a
2 **UnicodeCategory** enumerated constant.

3 *Return Value:* A **System.Globalization.UnicodeCategory** enumerated constant
4 that identifies the group that contains *c* . A Unicode character.

5 GetUnicodeCategory

6
7 [C#] public static UnicodeCategory GetUnicodeCategory(string s, int index);

8 [C++] public: static UnicodeCategory GetUnicodeCategory(String* s, int index);

9 [VB] Public Shared Function GetUnicodeCategory(ByVal s As String, ByVal
10 index As Integer) As UnicodeCategory

11 [JScript] public static function GetUnicodeCategory(s : String, index : int) :
12 UnicodeCategory;

13
14 *Description*

15 Categorizes the character at the specified position in a specified **String** into
16 a group identified by a **UnicodeCategory** enumerated constant.

17 *Return Value:* A **System.Globalization.UnicodeCategory** enumerated constant
18 that identifies the group that contains the character at position *index* in *s* .

19 Character positions in a **String** are indexed starting from zero. A

20 **System.String**. A 32-bit signed integer that specifies a character position in *s*.

21 IsControl

22
23 [C#] public static bool IsControl(char c);

24 [C++] public: static bool IsControl(__wchar_t c);

25 [VB] Public Shared Function IsControl(ByVal c As Char) As Boolean

[JScript] public static function IsControl(c : Char) : Boolean; Indicates whether a specified Unicode character is categorized as a control character.

Description

Indicates whether the specified Unicode character is categorized as a control character.

Return Value: **true** if *c* is a control character; otherwise, **false** . A Unicode character.

IsControl

[C#] public static bool IsControl(string s, int index);

[C++] public: static bool IsControl(String* s, int index);

[VB] Public Shared Function IsControl(ByVal s As String, ByVal index As Integer) As Boolean

[JScript] public static function IsControl(s : String, index : int) : Boolean;

Description

Indicates whether the character at the specified position in a specified **String** is categorized as a control character.

Return Value: **true** if the character at position *index* in *s* is a control character; otherwise, **false** .

Character positions in a **String** are indexed starting from zero. A **System.String**. A 32-bit signed integer that specifies a character position in *s*.

IsDigit

```

1
2 [C#] public static bool IsDigit(char c);
3 [C++] public: static bool IsDigit(__wchar_t c);
4 [VB] Public Shared Function IsDigit(ByVal c As Char) As Boolean
5 [JScript] public static function IsDigit(c : Char) : Boolean; Indicates whether a
6 Unicode character is categorized as a decimal digit.

```

8 *Description*

9 Indicates whether the specified Unicode character is categorized as a
10 decimal digit.

11 *Return Value:* **true** if *c* is a decimal digit; otherwise, **false** . A Unicode character.

12 *IsDigit*

```

13
14 [C#] public static bool IsDigit(string s, int index);
15 [C++] public: static bool IsDigit(String* s, int index);
16 [VB] Public Shared Function IsDigit(ByVal s As String, ByVal index As Integer)
17 As Boolean
18 [JScript] public static function IsDigit(s : String, index : int) : Boolean;
19

```

20 *Description*

21 Indicates whether the character at the specified position in a specified
22 **String** is categorized as a decimal digit.

23 *Return Value:* **true** if the character at position *index* in *s* is a decimal digit;
24 otherwise, **false** .

Character positions in a **String** are indexed starting from zero. A
System.String. A 32-bit signed integer that specifies a character position in *s*.

IsLetter

[C#] public static bool IsLetter(char c);

[C++] public: static bool IsLetter(__wchar_t c);

[VB] Public Shared Function IsLetter(ByVal c As Char) As Boolean

[JScript] public static function IsLetter(c : Char) : Boolean; Indicates whether a
Unicode character is categorized as an alphabetic letter.

Description

Indicates whether the specified Unicode character is categorized as an
alphabetic letter.

Return Value: **true** if *c* is an alphabetic letter; otherwise, **false** .

Valid letters are members of the following categories in
System.Globalization.UnicodeCategory : **UppercaseLetter** , **LowercaseLetter** ,
TitlecaseLetter , **ModifierLetter** , and **OtherLetter** . A Unicode character.

IsLetter

[C#] public static bool IsLetter(string s, int index);

[C++] public: static bool IsLetter(String* s, int index);

[VB] Public Shared Function IsLetter(ByVal s As String, ByVal index As Integer)
As Boolean

[JScript] public static function IsLetter(s : String, index : int) : Boolean;

1
2 *Description*

3 Indicates whether the character at the specified position in a specified
4 **String** is categorized as an alphabetic character.

5 *Return Value:* **true** if the character at position *index* in *s* is an alphabetic character;
6 otherwise, **false** .

7 Character positions in a **String** are indexed starting from zero. A
8 **System.String**. A 32-bit signed integer that specifies a character position in *s*.

9 **IsLetterOrDigit**

10
11 [C#] public static bool IsLetterOrDigit(char c);

12 [C++] public: static bool IsLetterOrDigit(__wchar_t c);

13 [VB] Public Shared Function IsLetterOrDigit(ByVal c As Char) As Boolean

14 [JScript] public static function IsLetterOrDigit(c : Char) : Boolean; Indicates
15 whether a Unicode character is categorized as a letter or decimal digit.

16
17 *Description*

18 Indicates whether the specified Unicode character is categorized as a letter
19 or decimal digit.

20 *Return Value:* **true** if *c* is a letter or decimal digit; otherwise, **false** . A Unicode
21 character.

22 **IsLetterOrDigit**

23
24 [C#] public static bool IsLetterOrDigit(string s, int index);

25 [C++] public: static bool IsLetterOrDigit(String* s, int index);

1 [VB] Public Shared Function IsLetterOrDigit(ByVal s As String, ByVal index As
2 Integer) As Boolean

3 [JScript] public static function IsLetterOrDigit(s : String, index : int) : Boolean;

4
5 *Description*

6 Indicates whether the character at the specified position in a specified
7 **String** is categorized as an alphabetic character or a decimal digit.

8 *Return Value:* **true** if the character at position *index* in *s* is an alphabetic character
9 or a decimal digit; otherwise, **false** .

10 Character positions in a **String** are indexed starting from zero. A
11 **System.String**. A 32-bit signed integer that specifies a character position in *s*.

12 **IsLower**

13
14 [C#] public static bool IsLower(char c);

15 [C++] public: static bool IsLower(__wchar_t c);

16 [VB] Public Shared Function IsLower(ByVal c As Char) As Boolean

17 [JScript] public static function IsLower(c : Char) : Boolean; Indicates whether a
18 Unicode character is categorized as a lowercase letter.

19
20 *Description*

21 Indicates whether the specified Unicode character is categorized as a
22 lowercase letter.

23 *Return Value:* **true** if *c* is a lowercase letter; otherwise, **false** . A Unicode
24 character.

25 **IsLower**

1
2 [C#] public static bool IsLower(string s, int index);

3 [C++] public: static bool IsLower(String* s, int index);

4 [VB] Public Shared Function IsLower(ByVal s As String, ByVal index As
5 Integer) As Boolean

6 [JScript] public static function IsLower(s : String, index : int) : Boolean;

7
8 *Description*

9 Indicates whether the character at the specified position in a specified
10 **String** is categorized as a lowercase letter.

11 *Return Value:* **true** if the character at position *index* in *s* is a lowercase letter;
12 otherwise, **false** .

13 Character positions in a **String** are indexed starting from zero. A
14 **System.String**. A 32-bit signed integer that specifies a character position in *s*.

15 **IsNumber**

16
17 [C#] public static bool IsNumber(char c);

18 [C++] public: static bool IsNumber(__wchar_t c);

19 [VB] Public Shared Function IsNumber(ByVal c As Char) As Boolean

20 [JScript] public static function IsNumber(c : Char) : Boolean; Indicates whether a
21 Unicode character is categorized as a decimal digit or hexadecimal number.

22
23 *Description*

24 Indicates whether the specified Unicode character is categorized as a
25 decimal digit or hexadecimal number.

1 *Return Value:* **true** if *c* is a decimal digit or hexadecimal number; otherwise, **false**

2 . A Unicode character.

3 IsNumber

4
5 [C#] public static bool IsNumber(string s, int index);

6 [C++] public: static bool IsNumber(String* s, int index);

7 [VB] Public Shared Function IsNumber(ByVal s As String, ByVal index As
8 Integer) As Boolean

9 [JScript] public static function IsNumber(s : String, index : int) : Boolean;

10
11 *Description*

12 Indicates whether the character at the specified position in a specified
13 **String** is categorized as a decimal digit or hexadecimal number.

14 *Return Value:* **true** if the character at position *index* in *s* is a decimal digit or
15 hexadecimal number; otherwise, **false** .

16 Character positions in a **String** are indexed starting from zero. A

17 **System.String**. A 32-bit signed integer that specifies a character position in *s*.

18 IsPunctuation

19
20 [C#] public static bool IsPunctuation(char c);

21 [C++] public: static bool IsPunctuation(__wchar_t c);

22 [VB] Public Shared Function IsPunctuation(ByVal c As Char) As Boolean

23 [JScript] public static function IsPunctuation(c : Char) : Boolean; Indicates
24 whether a Unicode character is categorized as a punctuation mark.

Description

Indicates whether the specified Unicode character is categorized as a punctuation mark.

Return Value: **true** if *c* is a punctuation mark; otherwise, **false** . A Unicode character.

IsPunctuation

[C#] public static bool IsPunctuation(string s, int index);

[C++] public: static bool IsPunctuation(String* s, int index);

[VB] Public Shared Function IsPunctuation(ByVal s As String, ByVal index As Integer) As Boolean

[JScript] public static function IsPunctuation(s : String, index : int) : Boolean;

Description

Indicates whether the character at the specified position in a specified **String** is categorized as a punctuation mark.

Return Value: **true** if the character at position *index* in *s* is a punctuation mark; otherwise, **false** .

Character positions in a **String** are indexed starting from zero. A **System.String**. A 32-bit signed integer that specifies a character position in *s*.

IsSeparator

[C#] public static bool IsSeparator(char c);

[C++] public: static bool IsSeparator(__wchar_t c);

1 [VB] Public Shared Function IsSeparator(ByVal c As Char) As Boolean

2 [JScript] public static function IsSeparator(c : Char) : Boolean; Indicates whether a
3 Unicode character is categorized as a separator character.

4
5 *Description*

6 Indicates whether the specified Unicode character is categorized as a
7 separator character.

8 *Return Value:* **true** if *c* is a separator character; otherwise, **false** . A Unicode
9 character.

10 **IsSeparator**

11
12 [C#] public static bool IsSeparator(string s, int index);

13 [C++] public: static bool IsSeparator(String* s, int index);

14 [VB] Public Shared Function IsSeparator(ByVal s As String, ByVal index As
15 Integer) As Boolean

16 [JScript] public static function IsSeparator(s : String, index : int) : Boolean;

17
18 *Description*

19 Indicates whether the character at the specified position in a specified
20 **String** is categorized as a separator character.

21 *Return Value:* **true** if the character at position *index* in *s* is a separator character;
22 otherwise, **false** .

23 Character positions in a **String** are indexed starting from zero. A

24 **System.String**. A 32-bit signed integer that specifies a character position in *s*.

25 **IsSurrogate**

1
2 [C#] public static bool IsSurrogate(char c);

3 [C++] public: static bool IsSurrogate(__wchar_t c);

4 [VB] Public Shared Function IsSurrogate(ByVal c As Char) As Boolean

5 [JScript] public static function IsSurrogate(c : Char) : Boolean; Indicates whether
6 a Unicode character is categorized as a surrogate character.

7
8 *Description*

9 Indicates whether the specified Unicode character is categorized as a
10 surrogate character.

11 *Return Value:* **true** if *c* is a surrogate character; otherwise, **false** .

12 For more information about surrogate pairs, see the Unicode Standard at
13 <http://www.unicode.org>. A Unicode character.

14 **IsSurrogate**

15
16 [C#] public static bool IsSurrogate(string s, int index);

17 [C++] public: static bool IsSurrogate(String* s, int index);

18 [VB] Public Shared Function IsSurrogate(ByVal s As String, ByVal index As
19 Integer) As Boolean

20 [JScript] public static function IsSurrogate(s : String, index : int) : Boolean;

21
22 *Description*

23 Indicates whether the character at the specified position in a specified
24 **String** is categorized as a surrogate character.

1 *Return Value:* **true** if the character at position *index* in *s* is a surrogate character;
2 otherwise, **false** .

3 Character positions in a **String** are indexed starting from zero. A

4 **System.String**. A 32-bit signed integer that specifies a character position in *s*.

5 **IsSymbol**

6
7 [C#] public static bool IsSymbol(char c);

8 [C++] public: static bool IsSymbol(__wchar_t c);

9 [VB] Public Shared Function IsSymbol(ByVal c As Char) As Boolean

10 [JScript] public static function IsSymbol(c : Char) : Boolean; Indicates whether a
11 Unicode character is categorized as a symbol character.

12
13 *Description*

14 Indicates whether the specified Unicode character is categorized as a
15 symbol character.

16 *Return Value:* **true** if *c* is a symbol character; otherwise, **false** .

17 Valid symbols are members of the following categories in

18 **System.Globalization.UnicodeCategory** : **MathSymbol** , **CurrencySymbol** ,

19 **ModifierSymbol** , and **OtherSymbol** . A Unicode character.

20 **IsSymbol**

21
22 [C#] public static bool IsSymbol(string s, int index);

23 [C++] public: static bool IsSymbol(String* s, int index);

24 [VB] Public Shared Function IsSymbol(ByVal s As String, ByVal index As
25 Integer) As Boolean

[JScript] public static function IsSymbol(s : String, index : int) : Boolean;

Description

Indicates whether the character at the specified position in a specified **String** is categorized as a symbol character.

Return Value: **true** if the character at position *index* in *s* is a symbol character; otherwise, **false** .

Character positions in a **String** are indexed starting from zero. A **System.String**. A 32-bit signed integer that specifies a character position in *s*.

IsUpper

[C#] public static bool IsUpper(char c);

[C++] public: static bool IsUpper(__wchar_t c);

[VB] Public Shared Function IsUpper(ByVal c As Char) As Boolean

[JScript] public static function IsUpper(c : Char) : Boolean; Indicates whether a Unicode character is categorized as an uppercase letter.

Description

Indicates whether the specified Unicode character is categorized as an uppercase letter.

Return Value: **true** if *c* is an uppercase letter; otherwise, **false** . A Unicode character.

IsUpper

[C#] public static bool IsUpper(string s, int index);

```

1 [C++] public: static bool IsUpper(String* s, int index);
2 [VB] Public Shared Function IsUpper(ByVal s As String, ByVal index As Integer)
3 As Boolean
4 [JScript] public static function IsUpper(s : String, index : int) : Boolean;
5

```

Description

Indicates whether the character at the specified position in a specified **String** is categorized as an uppercase letter.

Return Value: **true** if the character at position *index* in *s* is an uppercase letter; otherwise, **false** .

Character positions in a **String** are indexed starting from zero. A **System.String**. A 32-bit signed integer that specifies a character position in *s*.

IsWhiteSpace

```

15 [C#] public static bool IsWhiteSpace(char c);
16 [C++] public: static bool IsWhiteSpace(__wchar_t c);
17 [VB] Public Shared Function IsWhiteSpace(ByVal c As Char) As Boolean
18 [JScript] public static function IsWhiteSpace(c : Char) : Boolean; Indicates
19 whether a Unicode character is categorized as white space.
20

```

Description

Indicates whether the specified Unicode character is categorized as white space.

Return Value: **true** if *c* is white space; otherwise, **false** .

Valid white space characters are members of the **SpaceSeparator** category in **System.Globalization.UnicodeCategory** , as well as these Unicode characters: hexadecimal 0x0009, 0x000a, 0x000b, 0x000c, 0x000d, 0x0085, 0x2028, and 0x2029. A Unicode character.

IsWhiteSpace

```
[C#] public static bool IsWhiteSpace(string s, int index);  
[C++] public: static bool IsWhiteSpace(String* s, int index);  
[VB] Public Shared Function IsWhiteSpace(ByVal s As String, ByVal index As Integer) As Boolean  
[JScript] public static function IsWhiteSpace(s : String, index : int) : Boolean;
```

Description

Indicates whether the character at the specified position in a specified **String** is categorized as white space.

Return Value: **true** if the character at position *index* in *s* is white space; otherwise, **false** .

Character positions in a **String** are indexed starting from zero. A **System.String**. A 32-bit signed integer that specifies a character position in *s*.

Parse

```
[C#] public static char Parse(string s);  
[C++] public: static __wchar_t Parse(String* s);  
[VB] Public Shared Function Parse(ByVal s As String) As Char  
[JScript] public static function Parse(s : String) : Char;
```

1
2 *Description*

3 Converts the value of the specified **String** to its equivalent Unicode
4 character.

5 *Return Value:* A Unicode character equivalent to the sole character in *s* . A

6 **System.String** containing a single character or **null**.

7 IConvertible.ToBoolean

8
9 [C#] bool IConvertible.ToBoolean(IFormatProvider provider);

10 [C++] bool IConvertible::ToBoolean(IFormatProvider* provider);

11 [VB] Function ToBoolean(ByVal provider As IFormatProvider) As Boolean

12 Implements IConvertible.ToBoolean

13 [JScript] function IConvertible.ToBoolean(provider : IFormatProvider) : Boolean;

14 IConvertible.ToByte

15
16 [C#] byte IConvertible.ToByte(IFormatProvider provider);

17 [C++] unsigned char IConvertible::ToByte(IFormatProvider* provider);

18 [VB] Function ToByte(ByVal provider As IFormatProvider) As Byte Implements

19 IConvertible.ToByte

20 [JScript] function IConvertible.ToByte(provider : IFormatProvider) : Byte;

21 IConvertible.ToChar

22
23 [C#] char IConvertible.ToChar(IFormatProvider provider);

24 [C++] __wchar_t IConvertible::ToChar(IFormatProvider* provider);

25 [VB] Function ToChar(ByVal provider As IFormatProvider) As Char Implements

1 IConvertible.ToChar

2 [JScript] function IConvertible.ToChar(provider : IFormatProvider) : Char;

3 IConvertible.ToDateTime

4
5 [C#] DateTime IConvertible.ToDateTime(IFormatProvider provider);

6 [C++] DateTime IConvertible::ToDateTime(IFormatProvider* provider);

7 [VB] Function ToDateTime(ByVal provider As IFormatProvider) As DateTime

8 Implements IConvertible.ToDateTime

9 [JScript] function IConvertible.ToDateTime(provider : IFormatProvider) :

10 DateTime;

11 IConvertible.ToDecimal

12
13 [C#] decimal IConvertible.ToDecimal(IFormatProvider provider);

14 [C++] Decimal IConvertible::ToDecimal(IFormatProvider* provider);

15 [VB] Function ToDecimal(ByVal provider As IFormatProvider) As Decimal

16 Implements IConvertible.ToDecimal

17 [JScript] function IConvertible.ToDecimal(provider : IFormatProvider) : Decimal;

18 IConvertible.ToDouble

19
20 [C#] double IConvertible.ToDouble(IFormatProvider provider);

21 [C++] double IConvertible::ToDouble(IFormatProvider* provider);

22 [VB] Function ToDouble(ByVal provider As IFormatProvider) As Double

23 Implements IConvertible.ToDouble

24 [JScript] function IConvertible.ToDouble(provider : IFormatProvider) : double;

25 IConvertible.ToInt16

```

1
2 [C#] short IConvertible.ToInt16(IFormatProvider provider);
3 [C++] short IConvertible::ToInt16(IFormatProvider* provider);
4 [VB] Function ToInt16(ByVal provider As IFormatProvider) As Short
5 Implements IConvertible.ToInt16
6 [JScript] function IConvertible.ToInt16(provider : IFormatProvider) : Int16;
7     IConvertible.ToInt32
8
9 [C#] int IConvertible.ToInt32(IFormatProvider provider);
10 [C++] int IConvertible::ToInt32(IFormatProvider* provider);
11 [VB] Function ToInt32(ByVal provider As IFormatProvider) As Integer
12 Implements IConvertible.ToInt32
13 [JScript] function IConvertible.ToInt32(provider : IFormatProvider) : int;
14     IConvertible.ToInt64
15
16 [C#] long IConvertible.ToInt64(IFormatProvider provider);
17 [C++] __int64 IConvertible::ToInt64(IFormatProvider* provider);
18 [VB] Function ToInt64(ByVal provider As IFormatProvider) As Long Implements
19 IConvertible.ToInt64
20 [JScript] function IConvertible.ToInt64(provider : IFormatProvider) : long;
21     IConvertible.ToSByte
22
23 [C#] sbyte IConvertible.ToSByte(IFormatProvider provider);
24 [C++] char IConvertible::ToSByte(IFormatProvider* provider);
25 [VB] Function ToSByte(ByVal provider As IFormatProvider) As SByte

```

1 Implements IConvertible.ToSByte

2 [JScript] function IConvertible.ToSByte(provider : IFormatProvider) : SByte;

3 IConvertible.ToSingle

5 [C#] float IConvertible.ToSingle(IFormatProvider provider);

6 [C++] float IConvertible::ToSingle(IFormatProvider* provider);

7 [VB] Function ToSingle(ByVal provider As IFormatProvider) As Single

8 Implements IConvertible.ToSingle

9 [JScript] function IConvertible.ToSingle(provider : IFormatProvider) : float;

10 IConvertible.ToType

12 [C#] object IConvertible.ToType(Type type, IFormatProvider provider);

13 [C++] Object* IConvertible::ToType(Type* type, IFormatProvider* provider);

14 [VB] Function ToType(ByVal type As Type, ByVal provider As IFormatProvider)

15 As Object Implements IConvertible.ToType

16 [JScript] function IConvertible.ToType(type : Type, provider : IFormatProvider) :

17 Object;

18 IConvertible.ToUInt16

20 [C#] ushort IConvertible.ToUInt16(IFormatProvider provider);

21 [C++] unsigned short IConvertible::ToUInt16(IFormatProvider* provider);

22 [VB] Function ToUInt16(ByVal provider As IFormatProvider) As UInt16

23 Implements IConvertible.ToUInt16

24 [JScript] function IConvertible.ToUInt16(provider : IFormatProvider) : UInt16;

25 IConvertible.ToInt32


```

[C#] uint IConvertible.ToUInt32(IFormatProvider provider);
[C++] unsigned int IConvertible::ToUInt32(IFormatProvider* provider);
[VB] Function ToUInt32(ByVal provider As IFormatProvider) As UInt32
Implements IConvertible.ToUInt32
[JScript] function IConvertible.ToUInt32(provider : IFormatProvider) : UInt32;

    IConvertible.ToUInt64

[C#] ulong IConvertible.ToUInt64(IFormatProvider provider);
[C++] unsigned __int64 IConvertible::ToUInt64(IFormatProvider* provider);
[VB] Function ToUInt64(ByVal provider As IFormatProvider) As UInt64
Implements IConvertible.ToUInt64
[JScript] function IConvertible.ToUInt64(provider : IFormatProvider) : UInt64;

    ToLower

[C#] public static char ToLower(char c);
[C++] public: static __wchar_t ToLower(__wchar_t c);
[VB] Public Shared Function ToLower(ByVal c As Char) As Char
[JScript] public static function ToLower(c : Char) : Char;
    
```

Description

Converts the value of a specified Unicode character to its lowercase equivalent using specified culture-specific formatting information.

Return Value: The lowercase equivalent of *c* .

Formatting information is obtained from the current culture. A Unicode character.

ToLower

[C#] public static char ToLower(char c, CultureInfo culture);

[C++] public: static __wchar_t ToLower(__wchar_t c, CultureInfo* culture);

[VB] Public Shared Function ToLower(ByVal c As Char, ByVal culture As CultureInfo) As Char

[JScript] public static function ToLower(c : Char, culture : CultureInfo) : Char;

Converts the value of a Unicode character to its lowercase equivalent.

Description

Converts the value of a specified Unicode character to its lowercase equivalent using specified culture-specific formatting information.

Return Value: The lowercase equivalent of *c* , formatted according to *culture* .

Use **System.String.ToLower** to convert a string to lowercase. A Unicode character. A **System.Globalization.CultureInfo** object that supplies culture-specific formatting information, or **null**.

ToString

[C#] public override string ToString();

[C++] public: String* ToString();

[VB] Overrides Public Function ToString() As String

[JScript] public override function ToString() : String; Converts the value of this instance to its equivalent **String** representation.

1
2 *Description*

3 Converts the value of this instance to its equivalent **String** representation.

4 *Return Value:* The **System.String** representation of the value of this instance.

5 ToString

6
7 [C#] public static string ToString(char c);

8 [C++] public: static String* ToString(__wchar_t c);

9 [VB] Public Shared Function ToString(ByVal c As Char) As String

10 [JScript] public static function ToString(c : Char) : String;

11
12 *Description*

13 Converts the specified Unicode character to its equivalent **String**
14 representation.

15 *Return Value:* The **System.String** representation of the value of *c* . A Unicode
16 character.

17 ToString

18
19 [C#] public string ToString(IFormatProvider provider);

20 [C++] public: __sealed String* ToString(IFormatProvider* provider);

21 [VB] NotOverridable Public Function ToString(ByVal provider As
22 IFormatProvider) As String

23 [JScript] public function ToString(provider : IFormatProvider) : String;

24
25 *Description*

Converts the value of this instance to its equivalent **String** representation using the specified culture-specific format information.

Return Value: The **System.String** representation of the value of this instance as specified by *provider*.

provider is ignored; it does not participate in this operation. (Reserved) An **System.IFormatProvider** interface implementation that supplies culture-specific formatting information.

ToUpper

[C#] public static char ToUpper(char c);

[C++] public: static __wchar_t ToUpper(__wchar_t c);

[VB] Public Shared Function ToUpper(ByVal c As Char) As Char

[JScript] public static function ToUpper(c : Char) : Char;

Description

Converts the value of a specified Unicode character to its uppercase equivalent using specified culture-specific formatting information.

Return Value: The uppercase equivalent of *c*.

Formatting information is obtained from the current culture. A Unicode character.

ToUpper

[C#] public static char ToUpper(char c, CultureInfo culture);

[C++] public: static __wchar_t ToUpper(__wchar_t c, CultureInfo* culture);

[VB] Public Shared Function ToUpper(ByVal c As Char, ByVal culture As

CultureInfo) As Char

[JScript] public static function ToUpper(c : Char, culture : CultureInfo) : Char;

Converts the value of a Unicode character to its uppercase equivalent.

Description

Converts the value of a specified Unicode character to its uppercase equivalent using specified culture-specific formatting information.

Return Value: The uppercase equivalent of *c* , formatted according to *culture* .

Use **System.String.ToUpper** to convert a string to uppercase. A Unicode character. A **System.Globalization.CultureInfo** object that supplies culture-specific formatting information, or **null**.

CharEnumerator class (System)

ToUpper

Description

Supports iterating over a **System.String** and reading its individual characters.

A **System.CharEnumerator** provides read-only access to the characters in a referenced **System.String** object. For example, the **foreach** statement of the Microsoft Visual Basic and C# programming languages, which iterates through the elements of a collection, retrieves a **System.CharEnumerator** from an instance of **System.String** in order to iterate through the characters in that instance.

Current

ToUpper

```
[C#] public char Current {get;}
[C++] public: __property __wchar_t get_Current();
[VB] Public ReadOnly Property Current As Char
[JScript] public function get Current() : Char;
```

Description

Gets the character in the enumerated string currently indexed by this instance.

This property should only be invoked when the index maintained by this instance is valid, otherwise, an exception is thrown. The index is always invalid for an empty string ("").

Clone

```
[C#] public object Clone();
[C++] public: __sealed Object* Clone();
[VB] NotOverridable Public Function Clone() As Object
[JScript] public function Clone() : Object;
```

Description

Creates a copy of this instance.

Return Value: An **System.Object** that is a copy of this instance.

1 The return value is a copy of this instance of **System.CharEnumerator**
2 and its current state. This is useful for saving your state while iterating through a
3 **System.String** object.

4 MoveNext

5
6 [C#] public bool MoveNext();
7 [C++] public: __sealed bool MoveNext();
8 [VB] NotOverridable Public Function MoveNext() As Boolean
9 [JScript] public function MoveNext() : Boolean;

10 Description

11 Increments the index to the next character of the enumerated string.

12
13 *Return Value:* **true** if the index is successfully incremented and within the
14 enumerated string; otherwise, **false** .

15 The **System.CharEnumerator.MoveNext** method increments the index by
16 one. Call **System.CharEnumerator.MoveNext** after calling
17 **System.String.GetEnumerator** or **System.CharEnumerator.Reset** to increment
18 the current character position to the first character in the enumerated string. Check
19 that the return value is **true** to determine that the current character position is
20 valid.

21 Reset

22
23 [C#] public void Reset();
24 [C++] public: __sealed void Reset();
25 [VB] NotOverridable Public Sub Reset()

1 [JScript] public function Reset();

3 *Description*

4 Initializes the index to a position logically before the first character of the
5 enumerated string.

6 The index is set to the invalid state.

7 CLSCompliantAttribute class (System)

8 ToString

11 *Description*

12 Indicates whether a program element is compliant with the Common
13 Language Specification (CLS). This class cannot be inherited.

14 If no **System.CLSCompliantAttribute** is applied to a program element, by
15 default: The assembly is not CLS-compliant.

16 CLSCompliantAttribute

17 *Example Syntax:*

18 ToString

20 [C#] public CLSCompliantAttribute(bool isCompliant);

21 [C++] public: CLSCompliantAttribute(bool isCompliant);

22 [VB] Public Sub New(ByVal isCompliant As Boolean)

23 [JScript] public function CLSCompliantAttribute(isCompliant : Boolean);

25 *Description*

1 Initializes an instance of the **System.CLSCompliantAttribute** class with a
2 Boolean value indicating whether the indicated program element is CLS-
3 compliant. **true** if CLS-compliant; otherwise, **false**.

4 **IsCompliant**

5 **ToString**

6
7 [C#] public bool IsCompliant {get;}

8 [C++] public: __property bool get_IsCompliant();

9 [VB] Public ReadOnly Property IsCompliant As Boolean

10 [JScript] public function get IsCompliant() : Boolean;

11
12 *Description*

13 Gets the Boolean value indicating whether the indicated program element is
14 CLS-compliant.

15 **TypeId**

16 Console class (System)

17 **ToString**

18
19
20 *Description*

21 Represents the standard input, output, and error streams for console
22 applications.

23 The **System.Console** class provides basic support for applications that read
24 characters from, and write characters to, the console. If the console does not exist,
25

as in a Windows-based application, writes to the console are not displayed and no exception is raised.

Error

ToString

[C#] public static TextWriter Error {get;}

[C++] public: __property static TextWriter* get_Error();

[VB] Public Shared ReadOnly Property Error As TextWriter

[JScript] public static function get Error() : TextWriter;

Description

Gets the standard error output stream.

This property is set to the standard error stream by default. This property can be set to another stream with the

System.Console.SetError(System.IO.TextWriter) method.

In

ToString

[C#] public static TextReader In {get;}

[C++] public: __property static TextReader* get_In();

[VB] Public Shared ReadOnly Property In As TextReader

[JScript] public static function get In() : TextReader;

Description

Gets the standard input stream.

This property is set to the standard input stream by default. This property can be set to another stream with the **System.Console.SetIn(System.IO.TextReader)** method.

Out

ToString

[C#] public static TextWriter Out {get;}

[C++] public: __property static TextWriter* get_Out();

[VB] Public Shared ReadOnly Property Out As TextWriter

[JScript] public static function get Out() : TextWriter;

Description

Gets the standard output stream.

This property is set to the standard output stream by default. This property can be set to another stream with the

System.Console.SetOut(System.IO.TextWriter) method.

OpenStandardError

[C#] public static Stream OpenStandardError();

[C++] public: static Stream* OpenStandardError();

[VB] Public Shared Function OpenStandardError() As Stream

[JScript] public static function OpenStandardError() : Stream; Acquires the standard error stream.

Description

Acquires the standard error stream.

Return Value: A **System.IO.TextWriter** object that represents the standard error stream.

This method can be used to reacquire the standard error stream after it has been changed by the **System.Console.SetError(System.IO.TextWriter)** method.

OpenStandardError

[C#] public static Stream OpenStandardError(int bufferSize);

[C++] public: static Stream* OpenStandardError(int bufferSize);

[VB] Public Shared Function OpenStandardError(ByVal bufferSize As Integer)

As Stream

[JScript] public static function OpenStandardError(bufferSize : int) : Stream;

Description

Acquires the standard error stream, set to a specified buffer size.

Return Value: A **System.IO.TextWriter** object that represents the standard error stream.

This method can be used to reacquire the standard error stream after it has been changed by the **System.Console.SetError(System.IO.TextWriter)** method.

The internal stream buffer size.

OpenStandardInput

[C#] public static Stream OpenStandardInput();

[C++] public: static Stream* OpenStandardInput();

[VB] Public Shared Function OpenStandardInput() As Stream

1 [JScript] public static function OpenStandardInput() : Stream; Acquires the
2 standard input stream.

3
4 *Description*

5 Acquires the standard input stream.

6 *Return Value:* A **System.IO.TextReader** object that represents the standard input
7 stream.

8 This method can be used to reacquire the standard input stream after it has
9 been changed by the **System.Console.SetIn(System.IO.TextReader)** method.

10 OpenStandardInput

11
12 [C#] public static Stream OpenStandardInput(int bufferSize);

13 [C++] public: static Stream* OpenStandardInput(int bufferSize);

14 [VB] Public Shared Function OpenStandardInput(ByVal bufferSize As Integer)

15 As Stream

16 [JScript] public static function OpenStandardInput(bufferSize : int) : Stream;

17
18 *Description*

19 Acquires the standard input stream, set to a specified buffer size.

20 *Return Value:* A **System.IO.TextReader** object that represents the standard
21 output stream.

22 This method can be used to reacquire the standard output stream after it has
23 been changed by the **System.Console.SetIn(System.IO.TextReader)** method.

24 The internal stream buffer size.

25 OpenStandardOutput

[C#] public static Stream OpenStandardOutput();
 [C++] public: static Stream* OpenStandardOutput();
 [VB] Public Shared Function OpenStandardOutput() As Stream
 [JScript] public static function OpenStandardOutput() : Stream; Acquires the
 standard output stream.

Description

Acquires the standard output stream.

Return Value: A **System.IO.TextWriter** object that represents the standard output stream.

This method can be used to reacquire the standard output stream after it has been changed by the **System.Console.SetOut(System.IO.TextWriter)** method.

OpenStandardOutput

[C#] public static Stream OpenStandardOutput(int bufferSize);
 [C++] public: static Stream* OpenStandardOutput(int bufferSize);
 [VB] Public Shared Function OpenStandardOutput(ByVal bufferSize As Integer)
 As Stream
 [JScript] public static function OpenStandardOutput(bufferSize : int) : Stream;

Description

Acquires the standard output stream, set to a specified buffer size.

Return Value: A **System.IO.TextWriter** object that represents the standard output stream.

1 This method can be used to reacquire the standard output stream after it has
2 been changed by the **System.Console.SetOut(System.IO.TextWriter)** method.

3 The internal stream buffer size.

4 Read

5
6 [C#] public static int Read();

7 [C++] public: static int Read();

8 [VB] Public Shared Function Read() As Integer

9 [JScript] public static function Read() : int;

10 11 *Description*

12 Reads the next character from the standard input stream.

13 *Return Value:* The next character from the input stream, or negative one if no
14 more characters are available.

15 This method will not return until the read operation is terminated (for
16 example, by the user pressing the enter key). If data is available, the input stream
17 contains what the user entered, suffixed with a carriage-return character followed
18 by a linefeed character ("\r\n").

19 ReadLine

20
21 [C#] public static string ReadLine();

22 [C++] public: static String* ReadLine();

23 [VB] Public Shared Function ReadLine() As String

24 [JScript] public static function ReadLine() : String;

Description

Reads the next line of characters from the standard input stream.

Return Value: The next line from the input stream, or **null** if no more characters are available.

A line is defined as a sequence of characters followed by a carriage return ('r'), a line feed ('n'), or a carriage return immediately followed by a line feed ('r\n'). The string that is returned does not contain the terminating carriage return and/or line feed.

SetError

[C#] public static void SetError(TextWriter newError);

[C++] public: static void SetError(TextWriter* newError);

[VB] Public Shared Sub SetError(ByVal newError As TextWriter)

[JScript] public static function SetError(newError : TextWriter);

Description

Sets the **System.Console.Error** property to the specified output stream.

By default, the **System.Console.Error** property is set to the standard error output stream. A **System.IO.TextWriter** stream that is the new standard error output.

SetIn

[C#] public static void SetIn(TextReader newIn);

[C++] public: static void SetIn(TextReader* newIn);

1 [VB] Public Shared Sub SetIn(ByVal newIn As TextReader)

2 [JScript] public static function SetIn(newIn : TextReader);

3
4 *Description*

5 Sets the **System.Console.In** property to the specified input stream.

6 By default, the **System.Console.In** property is set to the standard input
7 stream. A **System.IO.TextReader** stream that is the new standard input.

8 SetOut

9
10 [C#] public static void SetOut(TextWriter newOut);

11 [C++] public: static void SetOut(TextWriter* newOut);

12 [VB] Public Shared Sub SetOut(ByVal newOut As TextWriter)

13 [JScript] public static function SetOut(newOut : TextWriter);

14
15 *Description*

16 Sets the **System.Console.Out** property to the specified output stream.

17 By default, the **System.Console.Out** property is set to the standard output
18 stream. A **System.IO.TextWriter** stream that is the new standard output.

19 Write

20
21 [C#] public static void Write(bool value);

22 [C++] public: static void Write(bool value);

23 [VB] Public Shared Sub Write(ByVal value As Boolean)

24 [JScript] public static function Write(value : Boolean);

1
2 *Description*

3 Writes the text representation of the specified Boolean value to the standard
4 output stream.

5 The text representation of *value* is produced by calling
6 **System.Boolean.ToString** . The value to write.

7 Write

8
9 [C#] public static void Write(char value);

10 [C++] public: static void Write(__wchar_t value);

11 [VB] Public Shared Sub Write(ByVal value As Char)

12 [JScript] public static function Write(value : Char);

13
14 *Description*

15 Writes the specified Unicode character value to the standard output stream.
16 The value to write.

17 Write

18
19 [C#] public static void Write(char[] buffer);

20 [C++] public: static void Write(__wchar_t buffer __gc[]);

21 [VB] Public Shared Sub Write(ByVal buffer() As Char)

22 [JScript] public static function Write(buffer : Char[]);

23
24 *Description*

Writes the specified array of Unicode characters to the standard output stream. A Unicode character array.

Write

[C#] public static void Write(decimal value);

[C++] public: static void Write(Decimal value);

[VB] Public Shared Sub Write(ByVal value As Decimal)

[JScript] public static function Write(value : Decimal);

Description

Writes the text representation of the specified **System.Decimal** value to the standard output stream.

The text representation of *value* is produced by calling **System.Decimal.ToString** . The value to write.

Write

[C#] public static void Write(double value);

[C++] public: static void Write(double value);

[VB] Public Shared Sub Write(ByVal value As Double)

[JScript] public static function Write(value : double);

Description

Writes the text representation of the specified double-precision floating point value to the standard output stream.

The text representation of *value* is produced by calling
System.Double.ToString . The value to write.

Write

[C#] public static void Write(int value);

[C++] public: static void Write(int value);

[VB] Public Shared Sub Write(ByVal value As Integer)

[JScript] public static function Write(value : int);

Description

Writes the text representation of the specified 32-bit signed integer value to
the standard output stream.

The text representation of *value* is produced by calling
System.Int32.ToString . The value to write.

Write

[C#] public static void Write(long value);

[C++] public: static void Write(__int64 value);

[VB] Public Shared Sub Write(ByVal value As Long)

[JScript] public static function Write(value : long);

Description

Writes the text representation of the specified 64-bit signed integer value to
the standard output stream.

The text representation of *value* is produced by calling
System.Int64.ToString . The value to write.

Write

[C#] public static void Write(object value);
[C++] public: static void Write(Object* value);
[VB] Public Shared Sub Write(ByVal value As Object)
[JScript] public static function Write(value : Object);

Description

Writes the text representation of the specified object to the standard output stream.

If *value* is **null** , nothing is written and no exception is thrown. Otherwise, the **ToString** method of *value* is called to produce its string representation, and the resulting string is written to the standard output stream. The value to write.

Write

[C#] public static void Write(float value);
[C++] public: static void Write(float value);
[VB] Public Shared Sub Write(ByVal value As Single)
[JScript] public static function Write(value : float);

Description

Writes the text representation of the specified single-precision floating point value to the standard output stream.

The text representation of *value* is produced by calling
System.Single.ToString . The value to write.

Write

```
[C#] public static void Write(string value);  
[C++] public: static void Write(String* value);  
[VB] Public Shared Sub Write(ByVal value As String)  
[JScript] public static function Write(value : String);
```

Description

Writes the specified string value to the standard output stream.
If value is **null** , nothing is written to the standard output stream. The value
to write.

Write

```
[C#] public static void Write(uint value);  
[C++] public: static void Write(unsigned int value);  
[VB] Public Shared Sub Write(ByVal value As UInt32)  
[JScript] public static function Write(value : UInt32);
```

Description

Writes the text representation of the specified 32-bit unsigned integer value
to the standard output stream.

The text representation of *value* is produced by calling
System.UInt32.ToString . The value to write.

Write

[C#] public static void Write(ulong value);
[C++] public: static void Write(unsigned __int64 value);
[VB] Public Shared Sub Write(ByVal value As UInt64)
[JScript] public static function Write(value : UInt64);

Description

Writes the text representation of the specified 64-bit unsigned integer value to the standard output stream.

The text representation of *value* is produced by calling **System.UInt64.ToString** . The value to write.

Write

[C#] public static void Write(string format, object arg0);
[C++] public: static void Write(String* format, Object* arg0);
[VB] Public Shared Sub Write(ByVal format As String, ByVal arg0 As Object)
[JScript] public static function Write(format : String, arg0 : Object); Writes the specified information to the standard output stream.

Description

Writes the specified object to the standard output stream using the specified format information.

This method uses the same semantics as
System.String.Format(System.String,System.Object) . Format string. Object to
write using *format*.

Write

[C#] public static void Write(string format, params object[] arg);

[C++] public: static void Write(String* format, Object* arg __gc[]);

[VB] Public Shared Sub Write(ByVal format As String, ByVal ParamArray arg()
As Object)

[JScript] public static function Write(format : String, arg : Object[]);

Description

Writes the specified array of objects to the standard output stream using the
specified format information.

This method uses the same semantics as

System.String.Format(System.String,System.Object) . Format string. An array
of objects to write using *format*.

Write

[C#] public static void Write(char[] buffer, int index, int count);

[C++] public: static void Write(__wchar_t buffer __gc[], int index, int count);

[VB] Public Shared Sub Write(ByVal buffer() As Char, ByVal index As Integer,
ByVal count As Integer)

[JScript] public static function Write(buffer : Char[], index : int, count : int);

Description

Writes the specified subarray of Unicode characters to the standard output stream.

This method writes *count* characters starting at position *index* of *buffer* to the standard output stream. An array of Unicode characters. The starting position in *buffer*. The number of characters to write.

Write

[C#] public static void Write(string format, object arg0, object arg1);

[C++] public: static void Write(String* format, Object* arg0, Object* arg1);

[VB] Public Shared Sub Write(ByVal format As String, ByVal arg0 As Object, ByVal arg1 As Object)

[JScript] public static function Write(format : String, arg0 : Object, arg1 : Object);

Description

Writes the specified objects to the standard output stream using the specified format information.

This method uses the same semantics as **System.String.Format(System.String,System.Object)** . Format string. First object to write using *format*. Second object to write using *format*.

Write

[C#] public static void Write(string format, object arg0, object arg1, object arg2);

[C++] public: static void Write(String* format, Object* arg0, Object* arg1,

Object* arg2);

[VB] Public Shared Sub Write(ByVal format As String, ByVal arg0 As Object,
ByVal arg1 As Object, ByVal arg2 As Object)

[JScript] public static function Write(format : String, arg0 : Object, arg1 : Object,
arg2 : Object);

Description

Writes the specified objects to the standard output stream using the
specified format information.

This method uses the same semantics as
System.String.Format(System.String,System.Object) . Format string. First
object to write using *format*. Second object to write using *format*. Third object to
write using *format*.

Write

[C++] public: static void Write(String* format, Object* arg0, Object* arg1,
Object* arg2, Object* arg3, ...);

WriteLine

[C#] public static void WriteLine();

[C++] public: static void WriteLine();

[VB] Public Shared Sub WriteLine()

[JScript] public static function WriteLine(); Writes the specified data, followed by
the current line terminator, to the standard output stream.

Description

Writes the current line terminator to the standard output stream.

The default line terminator is a string whose value is a carriage return followed by a line feed ("`\r\n`"). Change the line terminator by setting the **System.IO.TextWriter.NewLine** property of the **System.Console.Out** property to another string.

WriteLine

[C#] public static void WriteLine(bool value);

[C++] public: static void WriteLine(bool value);

[VB] Public Shared Sub WriteLine(ByVal value As Boolean)

[JScript] public static function WriteLine(value : Boolean);

Description

Writes the text representation of the specified Boolean value, followed by the current line terminator, to the standard output stream.

The text representation of *value* is produced by calling **System.Boolean.ToString** . The value to write.

WriteLine

[C#] public static void WriteLine(char value);

[C++] public: static void WriteLine(__wchar_t value);

[VB] Public Shared Sub WriteLine(ByVal value As Char)

[JScript] public static function WriteLine(value : Char);

Description

Writes the specified Unicode character, followed by the current line terminator, value to the standard output stream.

For more information about the line terminator, see the Remarks section of the **System.Console.WriteLine** method that takes no parameters. The value to write.

WriteLine

[C#] public static void WriteLine(char[] buffer);

[C++] public: static void WriteLine(__wchar_t buffer __gc[]);

[VB] Public Shared Sub WriteLine(ByVal buffer() As Char)

[JScript] public static function WriteLine(buffer : Char[]);

Description

Writes the specified array of Unicode characters, followed by the current line terminator, to the standard output stream.

For more information about the line terminator, see the Remarks section of the **System.Console.WriteLine** method that takes no parameters. A Unicode character array.

WriteLine

[C#] public static void WriteLine(decimal value);

[C++] public: static void WriteLine(Decimal value);

[VB] Public Shared Sub WriteLine(ByVal value As Decimal)

1 [JScript] public static function WriteLine(value : Decimal);

3 *Description*

4 Writes the text representation of the specified **System.Decimal** value,
5 followed by the current line terminator, to the standard output stream.

6 The text representation of *value* is produced by calling
7 **System.Decimal.ToString** . The value to write.

8 WriteLine

10 [C#] public static void WriteLine(double value);

11 [C++] public: static void WriteLine(double value);

12 [VB] Public Shared Sub WriteLine(ByVal value As Double)

13 [JScript] public static function WriteLine(value : double);

15 *Description*

16 Writes the text representation of the specified double-precision floating
17 point value, followed by the current line terminator, to the standard output stream.

18 The text representation of *value* is produced by calling
19 **System.Double.ToString** . The value to write.

20 WriteLine

22 [C#] public static void WriteLine(int value);

23 [C++] public: static void WriteLine(int value);

24 [VB] Public Shared Sub WriteLine(ByVal value As Integer)

25 [JScript] public static function WriteLine(value : int);

1
2 *Description*

3 Writes the text representation of the specified 32-bit signed integer value,
4 followed by the current line terminator, to the standard output stream.

5 The text representation of *value* is produced by calling
6 **System.Int32.ToString** . The value to write.

7 WriteLine

8
9 [C#] public static void WriteLine(long value);

10 [C++] public: static void WriteLine(__int64 value);

11 [VB] Public Shared Sub WriteLine(ByVal value As Long)

12 [JScript] public static function WriteLine(value : long);

13
14 *Description*

15 Writes the text representation of the specified 64-bit signed integer value,
16 followed by the current line terminator, to the standard output stream.

17 The text representation of *value* is produced by calling
18 **System.Int64.ToString** . The value to write.

19 WriteLine

20
21 [C#] public static void WriteLine(object value);

22 [C++] public: static void WriteLine(Object* value);

23 [VB] Public Shared Sub WriteLine(ByVal value As Object)

24 [JScript] public static function WriteLine(value : Object);

Description

Writes the text representation of the specified object, followed by the current line terminator, to the standard output stream.

If *value* is **null**, nothing is written and no exception is thrown. Otherwise, the **ToString** method of *value* is called to produce its string representation, and the resulting string is written to the standard output stream. The value to write.

WriteLine

[C#] public static void WriteLine(float value);

[C++] public: static void WriteLine(float value);

[VB] Public Shared Sub WriteLine(ByVal value As Single)

[JScript] public static function WriteLine(value : float);

Description

Writes the text representation of the specified single-precision floating point value, followed by the current line terminator, to the standard output stream.

The text representation of *value* is produced by calling **System.Single.ToString**. The value to write.

WriteLine

[C#] public static void WriteLine(string value);

[C++] public: static void WriteLine(String* value);

[VB] Public Shared Sub WriteLine(ByVal value As String)

[JScript] public static function WriteLine(value : String);

Description

Writes the specified string value, followed by the current line terminator, to the standard output stream.

If value is **null**, nothing is written to the standard output stream. The value to write.

WriteLine

[C#] public static void WriteLine(uint value);

[C++] public: static void WriteLine(unsigned int value);

[VB] Public Shared Sub WriteLine(ByVal value As UInt32)

[JScript] public static function WriteLine(value : UInt32);

Description

Writes the text representation of the specified 32-bit unsigned integer value, followed by the current line terminator, to the standard output stream.

The text representation of *value* is produced by calling **System.UInt32.ToString**. The value to write.

WriteLine

[C#] public static void WriteLine(ulong value);

[C++] public: static void WriteLine(unsigned __int64 value);

[VB] Public Shared Sub WriteLine(ByVal value As UInt64)

[JScript] public static function WriteLine(value : UInt64);

Description

Writes the text representation of the specified 64-bit unsigned integer value, followed by the current line terminator, to the standard output stream.

The text representation of *value* is produced by calling **System.UInt64.ToString** . The value to write.

WriteLine

[C#] public static void WriteLine(string format, object arg0);

[C++] public: static void WriteLine(String* format, Object* arg0);

[VB] Public Shared Sub WriteLine(ByVal format As String, ByVal arg0 As Object)

[JScript] public static function WriteLine(format : String, arg0 : Object);

Description

Writes the specified object, followed by the current line terminator, to the standard output stream using the specified format information.

This method uses the same semantics as

System.String.Format(System.String,System.Object) . Format string. Object to write using *format*.

WriteLine

[C#] public static void WriteLine(string format, params object[] arg);

[C++] public: static void WriteLine(String* format, Object* arg __gc[]);

[VB] Public Shared Sub WriteLine(ByVal format As String, ByVal ParamArray

1 arg() As Object)

2 [JScript] public static function WriteLine(format : String, arg : Object[]);

3
4 *Description*

5 Writes the specified array of objects, followed by the current line
6 terminator, to the standard output stream using the specified format information.

7 This method uses the same semantics as

8 **System.String.Format(System.String,System.Object)** . Format string. An array
9 of objects to write using *format*.

10 **WriteLine**

11
12 [C#] public static void WriteLine(char[] buffer, int index, int count);

13 [C++] public: static void WriteLine(__wchar_t buffer __gc[], int index, int count);

14 [VB] Public Shared Sub WriteLine(ByVal buffer() As Char, ByVal index As
15 Integer, ByVal count As Integer)

16 [JScript] public static function WriteLine(buffer : Char[], index : int, count : int);

17
18 *Description*

19 Writes the specified subarray of Unicode characters, followed by the
20 current line terminator, to the standard output stream.

21 This method writes *count* characters starting at position *index* of *buffer* to
22 the standard output stream. An array of Unicode characters. The starting position
23 in *buffer*. The number of characters to write.

24 **WriteLine**

```

1
2 [C#] public static void WriteLine(string format, object arg0, object arg1);
3 [C++] public: static void WriteLine(String* format, Object* arg0, Object* arg1);
4 [VB] Public Shared Sub WriteLine(ByVal format As String, ByVal arg0 As
5 Object, ByVal arg1 As Object)
6 [JScript] public static function WriteLine(format : String, arg0 : Object, arg1 :
7 Object);
8

```

Description

Writes the specified objects, followed by the current line terminator, to the standard output stream using the specified format information.

This method uses the same semantics as

System.String.Format(System.String, System.Object) . Format string. First object to write using *format*. Second object to write using *format*.

WriteLine

```

17 [C#] public static void WriteLine(string format, object arg0, object arg1, object
18 arg2);
19 [C++] public: static void WriteLine(String* format, Object* arg0, Object* arg1,
20 Object* arg2);
21 [VB] Public Shared Sub WriteLine(ByVal format As String, ByVal arg0 As
22 Object, ByVal arg1 As Object, ByVal arg2 As Object)
23 [JScript] public static function WriteLine(format : String, arg0 : Object, arg1 :
24 Object, arg2 : Object);
25

```

Description

Writes the specified objects, followed by the current line terminator, to the standard output stream using the specified format information.

This method uses the same semantics as **System.String.Format(System.String,System.Object)** . Format string. First object to write using *format*. Second object to write using *format*. Third object to write using *format*.

WriteLine

```
[C++] public: static void WriteLine(String* format, Object* arg0, Object* arg1, Object* arg2, Object* arg3, ...);
```

ContextBoundObject class (System)

WriteLine

Description

Defines the base class for all context-bound classes.

Objects that reside in a context and are bound to the context rules are called context-bound objects. A context is a set of properties or usage rules that define an environment where a collection of objects resides. The rules are enforced when the objects are entering or leaving a context. Objects that are not context-bound are called agile objects.

ContextBoundObject

Example Syntax:

WriteLine

[C#] protected ContextBoundObject();

[C++] protected: ContextBoundObject();

[VB] Protected Sub New()

[JScript] protected function ContextBoundObject();

ContextMarshalException class (System)

ToString

Description

The exception that is thrown when an attempt to marshal an object across a context boundary fails.

Objects can marshal by value or by reference. Any attempt to pass an instance of an unmarshallable type through a context boundary will result in a

System.ContextMarshalException .

ContextMarshalException

Example Syntax:

ToString

[C#] public ContextMarshalException();

[C++] public: ContextMarshalException();

[VB] Public Sub New()

[JScript] public function ContextMarshalException(); Initializes a new instance of the **System.ContextMarshalException** class.

Description

Initializes a new instance of the **System.ContextMarshalException** class with default properties.

The following table shows the initial property values for an instance of **System.ContextMarshalException**.

ContextMarshalException

Example Syntax:

ToString

[C#] public ContextMarshalException(string message);

[C++] public: ContextMarshalException(String* message);

[VB] Public Sub New(ByVal message As String)

[JScript] public function ContextMarshalException(message : String);

Description

Initializes a new instance of the **System.ContextMarshalException** class with a specified error message.

The following table shows the initial property values for an instance of **System.ContextMarshalException**. The error message that explains the reason for the exception.

ContextMarshalException

Example Syntax:

ToString

```

1
2 [C#] protected ContextMarshalException(SerializationInfo info,
3 StreamingContext context);
4 [C++] protected: ContextMarshalException(SerializationInfo* info,
5 StreamingContext context);
6 [VB] Protected Sub New(ByVal info As SerializationInfo, ByVal context As
7 StreamingContext)
8 [JScript] protected function ContextMarshalException(info : SerializationInfo,
9 context : StreamingContext);
10

```

Description

Initializes a new instance of the **System.ContextMarshalException** class with serialized data. The object that holds the serialized object data. The contextual information about the source or destination.

ContextMarshalException

Example Syntax:

ToString

```

19 [C#] public ContextMarshalException(string message, Exception inner);
20 [C++] public: ContextMarshalException(String* message, Exception* inner);
21 [VB] Public Sub New(ByVal message As String, ByVal inner As Exception)
22 [JScript] public function ContextMarshalException(message : String, inner :
23 Exception);
24

```

Description

1 Initializes a new instance of the **System.ContextMarshalException** class
2 with a specified error message and a reference to the inner exception that is the
3 root cause of this exception.

4 When an **Exception** *X* is thrown as a direct result of a previous exception *Y*,
5 the **System.Exception.InnerException** property of *X* should contain a reference
6 to *Y*. The **InnerException** property returns the same value as was passed into the
7 constructor, or **null** if the inner exception value was not supplied to the
8 constructor. The error message that explains the reason for the exception. An
9 instance of **System.Exception** that is the cause of the current **Exception**. If *inner*
10 is non-null, then the current **Exception** is raised in a catch block handling *inner*.

11 HelpLink

12 HResult

13 InnerException

14 Message

15 Source

16 StackTrace

17 TargetSite

18 ContextStaticAttribute class (System)

19 ToString

20
21
22 *Description*

23 Indicates that the value of a static field is unique for a particular context.
24
25

1 A static field marked with **System.ContextStaticAttribute** is not shared
2 between contexts. If the indicated static field is accessed on a different context, it
3 will contain a different value.

4 ContextStaticAttribute

5 *Example Syntax:*

6 ToString

7
8 [C#] public ContextStaticAttribute();

9 [C++] public: ContextStaticAttribute();

10 [VB] Public Sub New()

11 [JScript] public function ContextStaticAttribute();

12
13 *Description*

14 Initializes a new instance of the **System.ContextStaticAttribute** class.

15 TypeId

16 Convert class (System)

17 ToString

18
19
20 *Description*

21 Converts base data types to other base data types.

22 This class returns a base type that is equivalent to the value of a specified
23 type.

24 ToString

1
2 [C#] public static readonly object DBNull;

3 [C++] public: static Object* DBNull;

4 [VB] Public Shared ReadOnly DBNull As Object

5 [JScript] public static var DBNull : Object;

6
7 *Description*

8 A constant representing a database column absent of data; that is, database
9 null.

10 *ChangeType*

11
12 [C#] public static object ChangeType(object value, Type conversionType);

13 [C++] public: static Object* ChangeType(Object* value, Type* conversionType);

14 [VB] Public Shared Function ChangeType(ByVal value As Object, ByVal
15 conversionType As Type) As Object

16 [JScript] public static function ChangeType(value : Object, conversionType :
17 Type) : Object;

18
19 *Description*

20 Returns an **Object** with the specified **Type** and whose value is equivalent
21 to the specified object.

22 *Return Value:* An object whose **Type** is *conversionType* and whose value is
23 equivalent to *value* .

This method uses the current thread's culture for the conversion. An **System.Object** that implements the **System.IConvertible** interface. A

System.Type.

ChangeType

[C#] public static object ChangeType(object value, TypeCode typeCode);

[C++] public: static Object* ChangeType(Object* value, TypeCode typeCode);

[VB] Public Shared Function ChangeType(ByVal value As Object, ByVal

typeCode As TypeCode) As Object

[JScript] public static function ChangeType(value : Object, typeCode : TypeCode)

: Object; Returns an **Object** with a specified type and whose value is equivalent to a specified object.

Description

Returns an **Object** with the specified **TypeCode** and whose value is equivalent to the specified object.

Return Value: An object whose underlying **TypeCode** is *typeCode* and whose value is equivalent to *value* . An **System.Object** that implements the

System.IConvertible interface. A **System.TypeCode**

ChangeType

[C#] public static object ChangeType(object value, Type conversionType, IFormatProvider provider);

[C++] public: static Object* ChangeType(Object* value, Type* conversionType, IFormatProvider* provider);

```

1 [VB] Public Shared Function ChangeType(ByVal value As Object, ByVal
2 conversionType As Type, ByVal provider As IFormatProvider) As Object
3 [JScript] public static function ChangeType(value : Object, conversionType :
4 Type, provider : IFormatProvider) : Object;

```

Description

Returns an **Object** with the specified **Type** and whose value is equivalent to the specified object. A parameter supplies culture-specific formatting information.

Return Value: An object whose **Type** is *conversionType* and whose value is equivalent to *value* .

provider enables the user to specify culture-specific conversion information about the contents of *value* . For example, if *value* is a **String** that represents a number, *provider* could supply culture-specific information about the notation used to represent that number. An **System.Object** that implements the **System.IConvertible** interface. A **System.Type**. An **System.IFormatProvider** interface implementation that supplies culture-specific formatting information.

ChangeType

```

20 [C#] public static object ChangeType(object value, TypeCode typeCode,
21 IFormatProvider provider);
22 [C++] public: static Object* ChangeType(Object* value, TypeCode typeCode,
23 IFormatProvider* provider);
24 [VB] Public Shared Function ChangeType(ByVal value As Object, ByVal
25 typeCode As TypeCode, ByVal provider As IFormatProvider) As Object

```

```
1 [JScript] public static function ChangeType(value : Object, typeCode : TypeCode,
2 provider : IFormatProvider) : Object;
```

4 *Description*

5 Returns an **Object** with the specified **TypeCode** and whose value is
6 equivalent to the specified object. A parameter supplies culture-specific formatting
7 information.

8 *Return Value:* An object whose underlying **TypeCode** is *typeCode* and whose
9 value is equivalent to *value* .

10 *provider* enables the user to specify culture-specific conversion information
11 about the contents of *value* . For example, if *value* is a **String** that represents a
12 number, *provider* could supply culture-specific information about the notation
13 used to represent that number. An **System.Object** that implements the
14 **System.IConvertible** interface. A **System.TypeCode**. An
15 **System.IFormatProvider** interface implementation that supplies culture-specific
16 formatting information.

17 FromBase64CharArray

19 [C#] public static byte[] FromBase64CharArray(char[] inArray, int offset, int
20 length);

21 [C++] public: static unsigned char FromBase64CharArray(__wchar_t inArray
22 __gc[], int offset, int length) __gc[];

23 [VB] Public Shared Function FromBase64CharArray(ByVal inArray() As Char,
24 ByVal offset As Integer, ByVal length As Integer) As Byte()

25 [JScript] public static function FromBase64CharArray(inArray : Char[], offset :

1 int, length : int) : Byte[];

3 *Description*

4 Converts the specified subset of an array of Unicode characters consisting
5 of base 64 digits to an equivalent array of 8-bit unsigned integers. Parameters
6 specify the offset and number of elements in the input array.

7 *Return Value:* An array of 8-bit unsigned integers equivalent to *length* elements at
8 position *offset* in *inArray* .

9 The subset in *inArray* is composed of base 64 digits. The base 64 digits in
10 ascending order from zero are the uppercase characters 'A' to 'Z', lowercase
11 characters 'a' to 'z', numerals '0' to '9', and the symbols '+' and '/'. The valueless
12 character, '=', is used for trailing padding. A Unicode character array. A position
13 within *inArray*. The number of elements in *inArray* to convert.

14 FromBase64String

15
16 [C#] public static byte[] FromBase64String(string s);

17 [C++] public: static unsigned char FromBase64String(String* s) __gc[];

18 [VB] Public Shared Function FromBase64String(ByVal s As String) As Byte()

19 [JScript] public static function FromBase64String(s : String) : Byte[];

20
21 *Description*

22 Converts the specified **String** representation of a value consisting of base
23 64 digits to an equivalent array of 8-bit unsigned integers.

24 *Return Value:* An array of 8-bit unsigned integers equivalent to *s* .

s is composed of base 64 digits. The base 64 digits in ascending order from zero are the uppercase characters 'A' to 'Z', lowercase characters 'a' to 'z', numerals '0' to '9', and the symbols '+' and '/'. The valueless character, '=', is used for trailing padding. A **System.String**.

GetTypeCode

```
[C#] public static TypeCode GetTypeCode(object value);  
[C++] public: static TypeCode GetTypeCode(Object* value);  
[VB] Public Shared Function GetTypeCode(ByVal value As Object) As  
TypeCode  
[JScript] public static function GetTypeCode(value : Object) : TypeCode;
```

Description

Returns the **TypeCode** for the specified object.

Return Value: The **System.TypeCode** for *value* , or **System.TypeCode.Empty** if *value* is **null** . An **System.Object** that implements the **System.IConvertible** interface.

IsDBNull

```
[C#] public static bool IsDBNull(object value);  
[C++] public: static bool IsDBNull(Object* value);  
[VB] Public Shared Function IsDBNull(ByVal value As Object) As Boolean  
[JScript] public static function IsDBNull(value : Object) : Boolean;
```

Description

1 Returns an indication whether the specified object is of type **DBNull** .

2 *Return Value:* **true** if *value* is of type **System.TypeCode.DbNull** ; otherwise,
3 **false** . An object.

4 ToBase64CharArray

5
6 [C#] public static int ToBase64CharArray(byte[] inArray, int offsetIn, int length,
7 char[] outArray, int offsetOut);

8 [C++] public: static int ToBase64CharArray(unsigned char inArray __gc[], int
9 offsetIn, int length, __wchar_t outArray __gc[], int offsetOut);

10 [VB] Public Shared Function ToBase64CharArray(ByVal inArray() As Byte,
11 ByVal offsetIn As Integer, ByVal length As Integer, ByVal outArray() As Char,
12 ByVal offsetOut As Integer) As Integer

13 [JScript] public static function ToBase64CharArray(inArray : Byte[], offsetIn : int,
14 length : int, outArray : Char[], offsetOut : int) : int;

15
16 *Description*

17 Converts the value of a subset of an 8-bit unsigned integer array to an
18 equivalent subset of a Unicode character array consisting of base 64 digits.

19 Parameters specify the subsets as offsets of the input and output arrays and the
20 number of elements in the input array.

21 *Return Value:* A 32-bit signed integer containing the number of bytes in *outArray* .

22 The subset of *length* elements of *inArray* starting at position *offsetIn* , are
23 taken as a numeric value and converted to a subset of elements in *outArray*
24 starting at position *offsetOut* . The return value indicates the number of converted
25 elements in *outArray* . The subset of *outArray* consists of base 64 digits. An input

array of 8-bit unsigned integers. A position within *inArray*. The number of elements of *inArray* to convert. An output array of Unicode characters. A position within *outArray*.

ToBase64String

[C#] public static string ToBase64String(byte[] inArray);

[C++] public: static String* ToBase64String(unsigned char inArray __gc[]);

[VB] Public Shared Function ToBase64String(ByVal inArray() As Byte) As

String

[JScript] public static function ToBase64String(inArray : Byte[]) : String;

Converts the value of an array of 8-bit unsigned integers to its equivalent **String** representation consisting of base 64 digits.

Description

Converts the value of an array of 8-bit unsigned integers to its equivalent **String** representation consisting of base 64 digits.

Return Value: The **System.String** representation, in base 64, of the contents of *inArray*.

The elements of *inArray* are taken as a numeric value and converted to a **String** representation consisting of base 64 digits. An array of 8-bit unsigned integers.

ToBase64String

[C#] public static string ToBase64String(byte[] inArray, int offset, int length);

[C++] public: static String* ToBase64String(unsigned char inArray __gc[], int

offset, int length);

[VB] Public Shared Function ToBase64String(ByVal inArray() As Byte, ByVal
offset As Integer, ByVal length As Integer) As String
[JScript] public static function ToBase64String(inArray : Byte[], offset : int,
length : int) : String;

Description

Converts the value of a subset of an array of 8-bit unsigned integers to its equivalent **String** representation consisting of base 64 digits. Parameters specify the subset as an offset and number of elements in the array.

Return Value: The **System.String** representation in base 64 of *length* elements of *inArray* starting at position *offset* .

The elements of *inArray* are taken as a numeric value and converted to a **String** representation in base 64. An array of 8-bit unsigned integers. An offset in *inArray*. The number of elements of *inArray* to convert.

ToBoolean

[C#] public static bool ToBoolean(bool value);
[C++] public: static bool ToBoolean(bool value);
[VB] Public Shared Function ToBoolean(ByVal value As Boolean) As Boolean
[JScript] public static function ToBoolean(value : Boolean) : Boolean;

Description

Returns the specified Boolean value; no actual conversion is performed.

Return Value: *value* is returned unchanged. A Boolean.

ToBoolean

```
[C#] public static bool ToBoolean(byte value);  
[C++] public: static bool ToBoolean(unsigned char value);  
[VB] Public Shared Function ToBoolean(ByVal value As Byte) As Boolean  
[JScript] public static function ToBoolean(value : Byte) : Boolean;
```

Description

Converts the value of the specified 8-bit unsigned integer to an equivalent Boolean value.

Return Value: **true** if *value* is non-zero; otherwise, **false** . An 8-bit unsigned integer.

ToBoolean

```
[C#] public static bool ToBoolean(char value);  
[C++] public: static bool ToBoolean(__wchar_t value);  
[VB] Public Shared Function ToBoolean(ByVal value As Char) As Boolean  
[JScript] public static function ToBoolean(value : Char) : Boolean;
```

Description

Conversion from **Char** to Boolean is not supported.

Return Value: (None) Attempt to convert **Char** to Boolean. A Unicode character.

ToBoolean

```
[C#] public static bool ToBoolean(DateTime value);
```

[C++] public: static bool ToBoolean(DateTime value);

[VB] Public Shared Function ToBoolean(ByVal value As DateTime) As Boolean

[JScript] public static function ToBoolean(value : DateTime) : Boolean;

Description

Calling this method always throws **System.InvalidCastException** .

This method is reserved for future use. A **System.DateTime**.

ToBoolean

[C#] public static bool ToBoolean(decimal value);

[C++] public: static bool ToBoolean(Decimal value);

[VB] Public Shared Function ToBoolean(ByVal value As Decimal) As Boolean

[JScript] public static function ToBoolean(value : Decimal) : Boolean;

Description

Converts the value of the specified **Decimal** number to an equivalent

Boolean value.

Return Value: **true** if *value* is non-zero; otherwise, **false** .

Description

Converts the value of the specified **Decimal** number to an equivalent

Boolean value.

Return Value: **true** if *value* is non-zero; otherwise, **false** . A **System.Decimal** number.

ToBoolean

[C#] public static bool ToBoolean(double value);

[C++] public: static bool ToBoolean(double value);

[VB] Public Shared Function ToBoolean(ByVal value As Double) As Boolean

[JScript] public static function ToBoolean(value : double) : Boolean;

Description

Converts the value of the specified double-precision floating point number to an equivalent Boolean value.

Return Value: **true** if *value* is non-zero; otherwise, **false** . A double-precision floating point number.

ToBoolean

[C#] public static bool ToBoolean(short value);

[C++] public: static bool ToBoolean(short value);

[VB] Public Shared Function ToBoolean(ByVal value As Short) As Boolean

[JScript] public static function ToBoolean(value : Int16) : Boolean;

Description

Converts the value of the specified 16-bit signed integer to an equivalent Boolean value.

Return Value: **true** if *value* is non-zero; otherwise, **false** . A 16-bit signed integer.

ToBoolean

[C#] public static bool ToBoolean(int value);

1 [C++] public: static bool ToBoolean(int value);

2 [VB] Public Shared Function ToBoolean(ByVal value As Integer) As Boolean

3 [JScript] public static function ToBoolean(value : int) : Boolean;

4
5 *Description*

6 Converts the value of the specified 32-bit signed integer to an equivalent
7 Boolean value.

8 *Return Value:* **true** if *value* is non-zero; otherwise, **false** . A 32-bit signed integer.

9 ToBoolean

10
11 [C#] public static bool ToBoolean(long value);

12 [C++] public: static bool ToBoolean(__int64 value);

13 [VB] Public Shared Function ToBoolean(ByVal value As Long) As Boolean

14 [JScript] public static function ToBoolean(value : long) : Boolean;

15
16 *Description*

17 Converts the value of the specified 64-bit signed integer to an equivalent
18 Boolean value.

19 *Return Value:* **true** if *value* is non-zero; otherwise, **false** . A 64-bit signed integer.

20 ToBoolean

21
22 [C#] public static bool ToBoolean(object value);

23 [C++] public: static bool ToBoolean(Object* value);

24 [VB] Public Shared Function ToBoolean(ByVal value As Object) As Boolean

25 [JScript] public static function ToBoolean(value : Object) : Boolean; Converts a

specified value to an equivalent Boolean value.

Description

Converts the value of a specified **Object** to an equivalent Boolean value.

Return Value: **false** if *value* equals **null** . An **System.Object** that implements the **System.IConvertible** interface or **null**.

ToBoolean

[C#] public static bool ToBoolean(sbyte value);

[C++] public: static bool ToBoolean(char value);

[VB] Public Shared Function ToBoolean(ByVal value As SByte) As Boolean

[JScript] public static function ToBoolean(value : SByte) : Boolean;

Description

Converts the value of the specified 8-bit signed integer to an equivalent Boolean value.

Return Value: **true** if *value* is non-zero; otherwise, **false** . An 8-bit signed integer.

ToBoolean

[C#] public static bool ToBoolean(float value);

[C++] public: static bool ToBoolean(float value);

[VB] Public Shared Function ToBoolean(ByVal value As Single) As Boolean

[JScript] public static function ToBoolean(value : float) : Boolean;

Description

Converts the value of the specified single-precision floating point number to an equivalent Boolean value.

Return Value: **true** if *value* is non-zero; otherwise, **false** . A single-precision floating point number.

ToBoolean

[C#] public static bool ToBoolean(string value);

[C++] public: static bool ToBoolean(String* value);

[VB] Public Shared Function ToBoolean(ByVal value As String) As Boolean

[JScript] public static function ToBoolean(value : String) : Boolean;

Description

Converts the specified **String** representation of a logical value to its Boolean equivalent.

Return Value: **true** if *value* equals **System.Boolean.TrueString** , or **false** if *value* equals **System.Boolean.FalseString** . A **System.String** that contains the value of either **System.Boolean.TrueString** or **System.Boolean.FalseString**.

ToBoolean

[C#] public static bool ToBoolean(ushort value);

[C++] public: static bool ToBoolean(unsigned short value);

[VB] Public Shared Function ToBoolean(ByVal value As UInt16) As Boolean

[JScript] public static function ToBoolean(value : UInt16) : Boolean;

Description

Converts the value of the specified 16-bit unsigned integer to an equivalent Boolean value.

Return Value: **true** if *value* is non-zero; otherwise, **false** . A 16-bit unsigned integer.

ToBoolean

[C#] public static bool ToBoolean(uint value);

[C++] public: static bool ToBoolean(unsigned int value);

[VB] Public Shared Function ToBoolean(ByVal value As UInt32) As Boolean

[JScript] public static function ToBoolean(value : UInt32) : Boolean;

Description

Converts the value of the specified 32-bit unsigned integer to an equivalent Boolean value.

Return Value: **true** if *value* is non-zero; otherwise, **false** . A 32-bit unsigned integer.

ToBoolean

[C#] public static bool ToBoolean(ulong value);

[C++] public: static bool ToBoolean(unsigned __int64 value);

[VB] Public Shared Function ToBoolean(ByVal value As UInt64) As Boolean

[JScript] public static function ToBoolean(value : UInt64) : Boolean;

Description

Converts the value of the specified 64-bit unsigned integer to an equivalent Boolean value.

Return Value: **true** if *value* is non-zero; otherwise, **false** . A 64-bit unsigned integer.

ToBoolean

[C#] public static bool ToBoolean(object value, IFormatProvider provider);

[C++] public: static bool ToBoolean(Object* value, IFormatProvider* provider);

[VB] Public Shared Function ToBoolean(ByVal value As Object, ByVal provider As IFormatProvider) As Boolean

[JScript] public static function ToBoolean(value : Object, provider : IFormatProvider) : Boolean;

Description

Converts the value of the specified **Object** to an equivalent Boolean value using the specified culture-specific formatting information.

Return Value: **false** if *value* equals **null** .

provider enables the user to specify culture-specific conversion information about the contents of *value* . For example, if *value* is a **String** that represents a number, *provider* could supply culture-specific information about the notation used to represent that number. An **System.Object** that implements the **System.IConvertible** interface or **null**. An **System.IFormatProvider** interface implementation that supplies culture-specific formatting information.

ToBoolean

```

1
2 [C#] public static bool ToBoolean(string value, IFormatProvider provider);
3 [C++] public: static bool ToBoolean(String* value, IFormatProvider* provider);
4 [VB] Public Shared Function ToBoolean(ByVal value As String, ByVal provider
5 As IFormatProvider) As Boolean
6 [JScript] public static function ToBoolean(value : String, provider :
7 IFormatProvider) : Boolean;
8

```

9 *Description*

10 Converts the specified **String** representation of a logical value to its
11 Boolean equivalent using the specified culture-specific formatting information.

12 *Return Value:* **true** if *value* equals **System.Boolean.TrueString** , or **false** if *value*
13 equals **System.Boolean.FalseString** .

14 *provider* is ignored; it does not participate in this operation. A string that
15 contains the value of either **System.Boolean.TrueString** or
16 **System.Boolean.FalseString**. (Reserved) An **System.IFormatProvider** interface
17 implementation that supplies culture-specific formatting information.

18 *ToByte*

```

19
20 [C#] public static byte ToByte(bool value);
21 [C++] public: static unsigned char ToByte(bool value);
22 [VB] Public Shared Function ToByte(ByVal value As Boolean) As Byte
23 [JScript] public static function ToByte(value : Boolean) : Byte;
24

```

25 *Description*

Converts the value of the specified Boolean value to the equivalent 8-bit unsigned integer.

Return Value: The number 1 if *value* is **true** ; otherwise, 0. A Boolean value.

ToByte

[C#] public static byte ToByte(byte value);

[C++] public: static unsigned char ToByte(unsigned char value);

[VB] Public Shared Function ToByte(ByVal value As Byte) As Byte

[JScript] public static function ToByte(value : Byte) : Byte;

Description

Returns the specified 8-bit unsigned integer; no actual conversion is performed.

Return Value: *value* is returned unchanged. An 8-bit unsigned integer.

ToByte

[C#] public static byte ToByte(char value);

[C++] public: static unsigned char ToByte(__wchar_t value);

[VB] Public Shared Function ToByte(ByVal value As Char) As Byte

[JScript] public static function ToByte(value : Char) : Byte;

Description

Converts the value of the specified Unicode character to the equivalent 8-bit unsigned integer.

1 *Return Value:* The 8-bit unsigned integer equivalent to *value* . A Unicode
2 character.

3 ToByte

4
5 [C#] public static byte ToByte(DateTime value);

6 [C++] public: static unsigned char ToByte(DateTime value);

7 [VB] Public Shared Function ToByte(ByVal value As DateTime) As Byte

8 [JScript] public static function ToByte(value : DateTime) : Byte;

9
10 *Description*

11 Calling this method always throws **System.InvalidCastException** .

12 This method is reserved for future use. A **System.DateTime**.

13 ToByte

14
15 [C#] public static byte ToByte(decimal value);

16 [C++] public: static unsigned char ToByte(Decimal value);

17 [VB] Public Shared Function ToByte(ByVal value As Decimal) As Byte

18 [JScript] public static function ToByte(value : Decimal) : Byte;

19
20 *Description*

21 Converts the value of the specified **Decimal** number to an equivalent 8-bit
22 unsigned integer.

23 *Return Value:* *value* rounded to the nearest 8-bit signed integer. If *value* is halfway
24 between two whole numbers, the even number is returned; that is, 4.5 is converted
25 to 4, and 5.5 is converted to 6. A **System.Decimal** number.

ToByte

[C#] public static byte ToByte(double value);
[C++] public: static unsigned char ToByte(double value);
[VB] Public Shared Function ToByte(Byte value As Double) As Byte
[JScript] public static function ToByte(value : double) : Byte;

Description

Converts the value of the specified double-precision floating point number to an equivalent 8-bit unsigned integer.

Return Value: *value* rounded to the nearest 8-bit signed integer. If *value* is halfway between two whole numbers, the even number is returned; that is, 4.5 is converted to 4, and 5.5 is converted to 6. A double-precision floating point number.

ToByte

[C#] public static byte ToByte(short value);
[C++] public: static unsigned char ToByte(short value);
[VB] Public Shared Function ToByte(Byte value As Short) As Byte
[JScript] public static function ToByte(value : Int16) : Byte;

Description

Converts the value of the specified 16-bit signed integer to an equivalent 8-bit unsigned integer.

Return Value: An 8-bit unsigned integer equivalent to the value of *value* . A 16-bit signed integer.

ToByte

[C#] public static byte ToByte(int value);

[C++] public: static unsigned char ToByte(int value);

[VB] Public Shared Function ToByte(ByVal value As Integer) As Byte

[JScript] public static function ToByte(value : int) : Byte;

Description

Converts the value of the specified 32-bit signed integer to an equivalent 8-bit unsigned integer.

Return Value: An 8-bit unsigned integer equivalent to the value of *value* . A 32-bit signed integer.

ToByte

[C#] public static byte ToByte(long value);

[C++] public: static unsigned char ToByte(__int64 value);

[VB] Public Shared Function ToByte(ByVal value As Long) As Byte

[JScript] public static function ToByte(value : long) : Byte;

Description

Converts the value of the specified 64-bit signed integer to an equivalent 8-bit unsigned integer.

Return Value: An 8-bit unsigned integer equivalent to the value of *value* . A 64-bit signed integer.

ToByte

```

1  [C#] public static byte ToByte(object value);
2
3  [C++] public: static unsigned char ToByte(Object* value);
4
5  [VB] Public Shared Function ToByte(ByVal value As Object) As Byte
6
7  [JScript] public static function ToByte(value : Object) : Byte; Converts a specified
8  value to an 8-bit unsigned integer.
9

```

Description

Converts the value of the specified **Object** to an 8-bit unsigned integer.

Return Value: An 8-bit unsigned integer equivalent to the value of *value* , or zero if *value* is **null** .

The return value is the result of invoking the **IConvertible.ToByte** method of the underlying type of *value* . An **System.Object** that implements the **System.IConvertible** interface or **null**.

ToByte

```

17 [C#] public static byte ToByte(sbyte value);
18
19 [C++] public: static unsigned char ToByte(char value);
20
21 [VB] Public Shared Function ToByte(ByVal value As SByte) As Byte
22
23 [JScript] public static function ToByte(value : SByte) : Byte;
24

```

Description

Converts the value of the specified 8-bit signed integer to an equivalent 8-bit unsigned integer.

1 *Return Value:* An 8-bit unsigned integer equivalent to the value of *value* . An 8-bit
2 signed integer.

3 ToByte

4
5 [C#] public static byte ToByte(float value);

6 [C++] public: static unsigned char ToByte(float value);

7 [VB] Public Shared Function ToByte(ByVal value As Single) As Byte

8 [JScript] public static function ToByte(value : float) : Byte;

9
10 *Description*

11 Converts the value of the specified single-precision floating point number
12 to an equivalent 8-bit unsigned integer.

13 *Return Value:* *value* rounded to the nearest 8-bit signed integer. If *value* is halfway
14 between two whole numbers, the even number is returned; that is, 4.5 is converted
15 to 4, and 5.5 is converted to 6. A single-precision floating point number.

16 ToByte

17
18 [C#] public static byte ToByte(string value);

19 [C++] public: static unsigned char ToByte(String* value);

20 [VB] Public Shared Function ToByte(ByVal value As String) As Byte

21 [JScript] public static function ToByte(value : String) : Byte;

22
23 *Description*

24 Converts the specified **String** representation of a number to an equivalent
25 8-bit unsigned integer.

1 *Return Value:* An 8-bit unsigned integer equivalent to the value of *value* . A

2 **System.String** containing a number to convert.

3 ToByte

4
5 [C#] public static byte ToByte(ushort value);

6 [C++] public: static unsigned char ToByte(unsigned short value);

7 [VB] Public Shared Function ToByte(ByVal value As UInt16) As Byte

8 [JScript] public static function ToByte(value : UInt16) : Byte;

9
10 *Description*

11 Converts the value of the specified 16-bit unsigned integer to an equivalent
12 8-bit unsigned integer.

13 *Return Value:* An 8-bit unsigned integer equivalent to the value of *value* . A 16-bit
14 unsigned integer.

15 ToByte

16
17 [C#] public static byte ToByte(uint value);

18 [C++] public: static unsigned char ToByte(unsigned int value);

19 [VB] Public Shared Function ToByte(ByVal value As UInt32) As Byte

20 [JScript] public static function ToByte(value : UInt32) : Byte;

21
22 *Description*

23 Converts the value of the specified 32-bit unsigned integer to an equivalent
24 8-bit unsigned integer.

Return Value: An 8-bit unsigned integer equivalent to the value of *value* . A 32-bit unsigned integer.

ToByte

[C#] public static byte ToByte(ulong value);

[C++] public: static unsigned char ToByte(unsigned __int64 value);

[VB] Public Shared Function ToByte(Byte value As UInt64) As Byte

[JScript] public static function ToByte(value : UInt64) : Byte;

Description

Converts the value of the specified 64-bit unsigned integer to an equivalent 8-bit unsigned integer.

Return Value: An 8-bit unsigned integer equivalent to the value of *value* . A 64-bit unsigned integer.

ToByte

[C#] public static byte ToByte(object value, IFormatProvider provider);

[C++] public: static unsigned char ToByte(Object* value, IFormatProvider* provider);

[VB] Public Shared Function ToByte(Byte value As Object, Byte provider As IFormatProvider) As Byte

[JScript] public static function ToByte(value : Object, provider : IFormatProvider) : Byte;

Description

Converts the value of the specified **Object** to an 8-bit unsigned integer using the specified culture-specific formatting information.

Return Value: An 8-bit unsigned integer equivalent to the value of *value* , or zero if *value* is **null** .

provider enables the user to specify culture-specific conversion information about the contents of *value* . For example, if *value* is a **String** that represents a number, *provider* could supply culture-specific information about the notation used to represent that number. An **System.Object** that implements the **System.IConvertible** interface. An **System.IFormatProvider** interface implementation that supplies culture-specific formatting information.

ToByte

[C#] public static byte ToByte(string value, IFormatProvider provider);

[C++] public: static unsigned char ToByte(String* value, IFormatProvider* provider);

[VB] Public Shared Function ToByte(ByVal value As String, ByVal provider As IFormatProvider) As Byte

[JScript] public static function ToByte(value : String, provider : IFormatProvider) : Byte;

Description

Converts the specified **String** representation of a number to an equivalent 8-bit signed integer using specified culture-specific formatting information.

Return Value: An 8-bit signed integer equivalent to the value of *value* .

provider is an **IFormatProvider** instance that obtains a **System.Globalization.NumberFormatInfo** object. The **NumberFormatInfo** object provides culture-specific information about the format of *value* . If *provider* is **null** , the **NumberFormatInfo** for the current culture is used. A **System.String** containing a number to convert. An **System.IFormatProvider** interface implementation that supplies culture-specific formatting information.

ToByte

[C#] public static byte ToByte(string value, int fromBase);

[C++] public: static unsigned char ToByte(String* value, int fromBase);

[VB] Public Shared Function ToByte(ByVal value As String, ByVal fromBase As Integer) As Byte

[JScript] public static function ToByte(value : String, fromBase : int) : Byte;

Description

Converts the **String** representation of a number in a specified base to an equivalent 8-bit unsigned integer.

Return Value: An 8-bit unsigned integer equivalent to the number in *value* . A **System.String** containing a number. The base of the number in *value*, which must be 2, 8, 10, or 16.

ToChar

[C#] public static char ToChar(bool value);

[C++] public: static __wchar_t ToChar(bool value);

[VB] Public Shared Function ToChar(ByVal value As Boolean) As Char

1 [JScript] public static function ToChar(value : Boolean) : Char;

2
3 *Description*

4 Calling this method always throws **System.InvalidCastException** .

5 This method is reserved for future use. A **System.Boolean** value.

6 ToChar

7
8 [C#] public static char ToChar(byte value);

9 [C++] public: static __wchar_t ToChar(unsigned char value);

10 [VB] Public Shared Function ToChar(ByVal value As Byte) As Char

11 [JScript] public static function ToChar(value : Byte) : Char;

12
13 *Description*

14 Converts the value of the specified 8-bit unsigned integer to its equivalent
15 Unicode character.

16 *Return Value:* The Unicode character equivalent to the value of *value* . An 8-bit
17 unsigned integer.

18 ToChar

19
20 [C#] public static char ToChar(char value);

21 [C++] public: static __wchar_t ToChar(__wchar_t value);

22 [VB] Public Shared Function ToChar(ByVal value As Char) As Char

23 [JScript] public static function ToChar(value : Char) : Char;

24
25 *Description*

1 Returns the specified Unicode character value; no actual conversion is
2 performed.

3 *Return Value:* *value* is returned unchanged. A Unicode character.

4 ToChar

5
6 [C#] public static char ToChar(DateTime value);

7 [C++] public: static __wchar_t ToChar(DateTime value);

8 [VB] Public Shared Function ToChar(ByVal value As DateTime) As Char

9 [JScript] public static function ToChar(value : DateTime) : Char;

10
11 *Description*

12 Calling this method always throws **System.InvalidCastException** .

13 This method is reserved for future use. A **System.DateTime**.

14 ToChar

15
16 [C#] public static char ToChar(decimal value);

17 [C++] public: static __wchar_t ToChar(Decimal value);

18 [VB] Public Shared Function ToChar(ByVal value As Decimal) As Char

19 [JScript] public static function ToChar(value : Decimal) : Char;

20
21 *Description*

22 Converts the value of the specified **Decimal** number to its equivalent
23 Unicode character.

24 *Return Value:* A Unicode character equivalent to the value of *value* .

The return value is the result of invoking the **IConvertible.ToChar** method of the underlying type of *value* . An **System.Decimal** number.

ToChar

[C#] public static char ToChar(double value);

[C++] public: static __wchar_t ToChar(double value);

[VB] Public Shared Function ToChar(ByVal value As Double) As Char

[JScript] public static function ToChar(value : double) : Char;

Description

Converts the value of the specified double-precision floating point number to its equivalent Unicode character.

Return Value: The Unicode character equivalent to the value of *value* .

The return value is the result of invoking the **IConvertible.ToChar** method of the underlying type of *value* . A double-precision floating point number.

ToChar

[C#] public static char ToChar(short value);

[C++] public: static __wchar_t ToChar(short value);

[VB] Public Shared Function ToChar(ByVal value As Short) As Char

[JScript] public static function ToChar(value : Int16) : Char;

Description

Converts the value of the specified 16-bit signed integer to its equivalent Unicode character.

Return Value: The Unicode character equivalent to the value of *value* . A 16-bit signed integer.

ToChar

[C#] public static char ToChar(int value);

[C++] public: static __wchar_t ToChar(int value);

[VB] Public Shared Function ToChar(ByVal value As Integer) As Char

[JScript] public static function ToChar(value : int) : Char;

Description

Converts the value of the specified 32-bit signed integer to its equivalent Unicode character.

Return Value: The Unicode character equivalent to the value of *value* . A 32-bit signed integer.

ToChar

[C#] public static char ToChar(long value);

[C++] public: static __wchar_t ToChar(__int64 value);

[VB] Public Shared Function ToChar(ByVal value As Long) As Char

[JScript] public static function ToChar(value : long) : Char;

Description

Converts the value of the specified 64-bit signed integer to its equivalent Unicode character.

Return Value: The Unicode character equivalent to the value of *value* . A 64-bit signed integer.

ToChar

[C#] public static char ToChar(object value);

[C++] public: static __wchar_t ToChar(Object* value);

[VB] Public Shared Function ToChar(ByVal value As Object) As Char

[JScript] public static function ToChar(value : Object) : Char; Converts a specified value to a Unicode character.

Description

Converts the value of the specified **Object** to a Unicode character.

Return Value: The Unicode character equivalent to the value of *value* .

The return value is the result of invoking the **IConvertible.ToChar** method of the underlying type of *value* . An **System.Object** that implements the **System.IConvertible** interface.

ToChar

[C#] public static char ToChar(sbyte value);

[C++] public: static __wchar_t ToChar(char value);

[VB] Public Shared Function ToChar(ByVal value As SByte) As Char

[JScript] public static function ToChar(value : SByte) : Char;

Description

Converts the value of the specified 8-bit signed integer to its equivalent Unicode character.

Return Value: The Unicode character equivalent to the value of *value* . An 8-bit signed integer.

ToChar

[C#] public static char ToChar(float value);

[C++] public: static __wchar_t ToChar(float value);

[VB] Public Shared Function ToChar(ByVal value As Single) As Char

[JScript] public static function ToChar(value : float) : Char;

Description

Converts the value of the specified single-precision floating point number to its equivalent Unicode character.

Return Value: The Unicode character equivalent to the value of *value* .

The return value is the result of invoking the **IConvertible.ToChar** method of the underlying type of *value* . An single-precision floating point number.

ToChar

[C#] public static char ToChar(string value);

[C++] public: static __wchar_t ToChar(String* value);

[VB] Public Shared Function ToChar(ByVal value As String) As Char

[JScript] public static function ToChar(value : String) : Char;

Description

Converts the first character of a **String** to a Unicode character.

Return Value: The Unicode character equivalent to the first and only character in *value.value* is **null** .

value must be **null** or a **String** containing a single character. A

System.String of length 1 or **null**.

ToChar

[C#] public static char ToChar(ushort value);

[C++] public: static __wchar_t ToChar(unsigned short value);

[VB] Public Shared Function ToChar(ByVal value As UInt16) As Char

[JScript] public static function ToChar(value : UInt16) : Char;

Description

Converts the value of the specified 16-bit unsigned integer to its equivalent Unicode character.

Return Value: The Unicode character equivalent to the value of *value* . A 16-bit unsigned integer.

ToChar

[C#] public static char ToChar(uint value);

[C++] public: static __wchar_t ToChar(unsigned int value);

[VB] Public Shared Function ToChar(ByVal value As UInt32) As Char

[JScript] public static function ToChar(value : UInt32) : Char;

Description

Converts the value of the specified 32-bit unsigned integer to its equivalent Unicode character.

Return Value: The Unicode character equivalent to the value of *value* . A 32-bit unsigned integer.

ToChar

[C#] public static char ToChar(ulong value);

[C++] public: static __wchar_t ToChar(unsigned __int64 value);

[VB] Public Shared Function ToChar(ByVal value As UInt64) As Char

[JScript] public static function ToChar(value : UInt64) : Char;

Description

Converts the value of the specified 64-bit unsigned integer to its equivalent Unicode character.

Return Value: The Unicode character equivalent to the value of *value* . A 64-bit unsigned integer.

ToChar

[C#] public static char ToChar(object value, IFormatProvider provider);

[C++] public: static __wchar_t ToChar(Object* value, IFormatProvider* provider);

[VB] Public Shared Function ToChar(ByVal value As Object, ByVal provider As IFormatProvider) As Char

[JScript] public static function ToChar(value : Object, provider : IFormatProvider) : Char;

Description

Converts the value of the specified **Object** to its equivalent Unicode character using the specified culture-specific formatting information.

Return Value: The Unicode character equivalent to the value of *value* .

The return value is the result of invoking the **IConvertible.ToChar** method of the underlying type of *value* . An **System.Object** that implements the **System.IConvertible** interface. An **System.IFormatProvider** interface implementation that supplies culture-specific formatting information.

ToChar

```
[C#] public static char ToChar(string value, IFormatProvider provider);
```

```
[C++] public: static __wchar_t ToChar(String* value, IFormatProvider*  
provider);
```

```
[VB] Public Shared Function ToChar(ByVal value As String, ByVal provider As  
IFormatProvider) As Char
```

```
[JScript] public static function ToChar(value : String, provider : IFormatProvider)  
: Char;
```

Description

Converts the first character of a **String** to a Unicode character using specified culture-specific formatting information.

Return Value: The Unicode character equivalent to the first and only character in *value.value* is **null** .

value must be **null** or a **String** containing a single character. A

System.String of length 1 or **null**. (Reserved) An **System.IFormatProvider** interface implementation that supplies culture-specific formatting information.

ToDateTime

[C#] public static DateTime.ToDateTime(bool value);

[C++] public: static DateTime.ToDateTime(bool value);

[VB] Public Shared Function.ToDateTime(ByVal value As Boolean) As
DateTime

[JScript] public static function.ToDateTime(value : Boolean) : DateTime;

Description

Calling this method always throws **System.InvalidCastException** .

This method is reserved for future use. A Boolean value.

ToDateTime

[C#] public static DateTime.ToDateTime(byte value);

[C++] public: static DateTime.ToDateTime(unsigned char value);

[VB] Public Shared Function.ToDateTime(ByVal value As Byte) As DateTime

[JScript] public static function.ToDateTime(value : Byte) : DateTime;

Description

Calling this method always throws **System.InvalidCastException** .

This method is reserved for future use. An 8-bit unsigned integer.

ToDateTime

[C#] public static DateTime ToDateTime(char value);

[C++] public: static DateTime ToDateTime(__wchar_t value);

[VB] Public Shared Function ToDateTime(ByVal value As Char) As DateTime

[JScript] public static function ToDateTime(value : Char) : DateTime;

Description

Calling this method always throws **System.InvalidCastException**.

This method is reserved for future use. A Unicode character.

DateTime

[C#] public static DateTime ToDateTime(DateTime value);

[C++] public: static DateTime ToDateTime(DateTime value);

[VB] Public Shared Function ToDateTime(ByVal value As DateTime) As

DateTime

[JScript] public static function ToDateTime(value : DateTime) : DateTime;

Converts a specified value to a **DateTime**.

Description

Returns the specified **DateTime**; no actual conversion is performed. A

System.DateTime.

DateTime

[C#] public static DateTime ToDateTime(decimal value);

[C++] public: static DateTime ToDateTime(Decimal value);

1 [VB] Public Shared Function ToDateTime(ByVal value As Decimal) As
2 DateTime

3 [JScript] public static function ToDateTime(value : Decimal) : DateTime;

4
5 *Description*

6 Calling this method always throws **System.InvalidCastException** .

7 This method is reserved for future use. A **System.Decimal** value.

8 ToDateTime

9
10 [C#] public static DateTime ToDateTime(double value);

11 [C++] public: static DateTime ToDateTime(double value);

12 [VB] Public Shared Function ToDateTime(ByVal value As Double) As DateTime

13 [JScript] public static function ToDateTime(value : double) : DateTime;

14
15 *Description*

16 Calling this method always throws **System.InvalidCastException** .

17 This method is reserved for future use. A double-precision floating point
18 value.

19 ToDateTime

20
21 [C#] public static DateTime ToDateTime(short value);

22 [C++] public: static DateTime ToDateTime(short value);

23 [VB] Public Shared Function ToDateTime(ByVal value As Short) As DateTime

24 [JScript] public static function ToDateTime(value : Int16) : DateTime;

Description

Calling this method always throws **System.InvalidCastException** .

This method is reserved for future use. A 16-bit signed integer.

ToDateTime

[C#] public static DateTime.ToDateTime(int value);

[C++] public: static DateTime.ToDateTime(int value);

[VB] Public Shared Function ToDateTime(ByVal value As Integer) As DateTime

[JScript] public static function ToDateTime(value : int) : DateTime;

Description

Calling this method always throws **System.InvalidCastException** .

This method is reserved for future use. A 32-bit signed integer.

ToDateTime

[C#] public static DateTime.ToDateTime(long value);

[C++] public: static DateTime.ToDateTime(__int64 value);

[VB] Public Shared Function ToDateTime(ByVal value As Long) As DateTime

[JScript] public static function ToDateTime(value : long) : DateTime;

Description

Calling this method always throws **System.InvalidCastException** .

This method is reserved for future use. A 64-bit signed integer.

ToDateTime

[C#] public static DateTime ToDateTime(object value);
 [C++] public: static DateTime ToDateTime(Object* value);
 [VB] Public Shared Function ToDateTime(ByVal value As Object) As DateTime
 [JScript] public static function ToDateTime(value : Object) : DateTime; Converts
 a specified value to a **DateTime** .

Description

Converts the value of the specified **Object** to a **DateTime** .

Return Value: A **DateTime** equivalent to the value of *value* , or zero if *value* is **null** .

The return value is the result of invoking the **IConvertible.ToDateTime** method of the underlying type of *value* . An **System.Object** that implements the **System.IConvertible** interface or **null**.

ToDateTime

[C#] public static DateTime ToDateTime(sbyte value);
 [C++] public: static DateTime ToDateTime(char value);
 [VB] Public Shared Function ToDateTime(ByVal value As SByte) As DateTime
 [JScript] public static function ToDateTime(value : SByte) : DateTime;

Description

Calling this method always throws **System.InvalidCastException** .

This method is reserved for future use. An 8-bit signed integer.

ToDateTime

1
2 [C#] public static DateTime ToDateTime(float value);

3 [C++] public: static DateTime ToDateTime(float value);

4 [VB] Public Shared Function ToDateTime(ByVal value As Single) As DateTime

5 [JScript] public static function ToDateTime(value : float) : DateTime;

6
7 *Description*

8 Calling this method always throws **System.InvalidCastException** .

9 This method is reserved for future use. A single-precision floating point
10 value.

11 ToDateTime

12
13 [C#] public static DateTime ToDateTime(string value);

14 [C++] public: static DateTime ToDateTime(String* value);

15 [VB] Public Shared Function ToDateTime(ByVal value As String) As DateTime

16 [JScript] public static function ToDateTime(value : String) : DateTime;

17
18 *Description*

19 Converts the specified **String** representation of a date and time to an
20 equivalent **DateTime** .

21 *Return Value:* A **DateTime** equivalent to the value of *value* .

22 The return value is the result of invoking the
23 **System.DateTime.Parse(System.String)** method on *value* . A **System.String**
24 containing a date and time to convert.

25 ToDateTime

```

1 [C#] public static DateTime ToDateTime(ushort value);
2
3 [C++] public: static DateTime ToDateTime(unsigned short value);
4
5 [VB] Public Shared Function ToDateTime(ByVal value As UInt16) As DateTime
6
7 [JScript] public static function ToDateTime(value : UInt16) : DateTime;
8

```

Description

Calling this method always throws **System.InvalidCastException** .

This method is reserved for future use. A 16-bit unsigned integer.

DateTime

```

9
10
11
12 [C#] public static DateTime ToDateTime(uint value);
13
14 [C++] public: static DateTime ToDateTime(unsigned int value);
15
16 [VB] Public Shared Function ToDateTime(ByVal value As UInt32) As DateTime
17
18 [JScript] public static function ToDateTime(value : UInt32) : DateTime;
19

```

Description

Calling this method always throws **System.InvalidCastException** .

This method is reserved for future use. A 32-bit unsigned integer.

DateTime

```

20
21
22 [C#] public static DateTime ToDateTime(ulong value);
23
24 [C++] public: static DateTime ToDateTime(unsigned __int64 value);
25
26 [VB] Public Shared Function ToDateTime(ByVal value As UInt64) As DateTime
27
28 [JScript] public static function ToDateTime(value : UInt64) : DateTime;
29

```

1
2 *Description*

3 Calling this method always throws **System.InvalidCastException** .

4 This method is reserved for future use. A 64-bit unsigned integer.

5 **DateTime**

6
7 [C#] public static DateTime ToDateTime(object value, IFormatProvider provider);

8 [C++] public: static DateTime ToDateTime(Object* value, IFormatProvider*
9 provider);

10 [VB] Public Shared Function ToDateTime(ByVal value As Object, ByVal
11 provider As IFormatProvider) As DateTime

12 [JScript] public static function ToDateTime(value : Object, provider :
13 IFormatProvider) : DateTime;

14
15 *Description*

16 Converts the value of the specified **Object** to a **DateTime** using the
17 specified culture-specific formatting information.

18 *Return Value:* A **DateTime** equivalent to the value of *value* , or zero if *value* is
19 **null** .

20 The return value is the result of invoking the **IConvertible.ToDateTime**
21 method of the underlying type of *value* . An **System.Object** that implements the
22 **System.IConvertible** interface. An **System.IFormatProvider** interface
23 implementation that supplies culture-specific formatting information.

24 **DateTime**

```

1
2 [C#] public static DateTime ToDateTime(string value, IFormatProvider provider);
3 [C++] public: static DateTime ToDateTime(String* value, IFormatProvider*
4 provider);
5 [VB] Public Shared Function ToDateTime(ByVal value As String, ByVal provider
6 As IFormatProvider) As DateTime
7 [JScript] public static function ToDateTime(value : String, provider :
8 IFormatProvider) : DateTime;
9

```

Description

Converts the specified **String** representation of a number to an equivalent **DateTime** using the specified culture-specific formatting information.

Return Value: A **DateTime** equivalent to the value of *value* .

The return value is the result of invoking the **System.DateTime.Parse(System.String)** method on *value* . A **System.String** containing a number to convert. An **System.IFormatProvider** interface implementation that supplies culture-specific formatting information.

ToDecimal

```

18
19
20 [C#] public static decimal ToDecimal(bool value);
21 [C++] public: static Decimal ToDecimal(bool value);
22 [VB] Public Shared Function ToDecimal(ByVal value As Boolean) As Decimal
23 [JScript] public static function ToDecimal(value : Boolean) : Decimal;
24

```

Description

Converts the value of the specified Boolean value to the equivalent

Decimal number.

Return Value: The number 1 if *value* is **true** ; otherwise, 0. A Boolean value.

ToDecimal

[C#] public static decimal ToDecimal(byte value);

[C++] public: static Decimal ToDecimal(unsigned char value);

[VB] Public Shared Function ToDecimal(ByVal value As Byte) As Decimal

[JScript] public static function ToDecimal(value : Byte) : Decimal;

Description

Converts the value of the specified 8-bit unsigned integer to the equivalent

Decimal number.

Return Value: The **Decimal** number equivalent to the value of *value* . An 8-bit unsigned integer.

ToDecimal

[C#] public static decimal ToDecimal(char value);

[C++] public: static Decimal ToDecimal(__wchar_t value);

[VB] Public Shared Function ToDecimal(ByVal value As Char) As Decimal

[JScript] public static function ToDecimal(value : Char) : Decimal;

Description

Calling this method always throws **System.InvalidCastException** .

This method is reserved for future use. A Unicode character.

ToDecimal

```
[C#] public static decimal ToDecimal(DateTime value);  
[C++] public: static Decimal ToDecimal(DateTime value);  
[VB] Public Shared Function ToDecimal(ByVal value As DateTime) As Decimal  
[JScript] public static function ToDecimal(value : DateTime) : Decimal;
```

Description

Calling this method always throws **System.InvalidCastException**.

This method is reserved for future use. A **System.DateTime**.

ToDecimal

```
[C#] public static decimal ToDecimal(decimal value);  
[C++] public: static Decimal ToDecimal(Decimal value);  
[VB] Public Shared Function ToDecimal(ByVal value As Decimal) As Decimal  
[JScript] public static function ToDecimal(value : Decimal) : Decimal;
```

Description

Returns the specified **Decimal** number; no actual conversion is performed.

Return Value: *value* is returned unchanged. A **Decimal** number.

ToDecimal

```
[C#] public static decimal ToDecimal(double value);  
[C++] public: static Decimal ToDecimal(double value);  
[VB] Public Shared Function ToDecimal(ByVal value As Double) As Decimal
```

1 [JScript] public static function ToDecimal(value : double) : Decimal;

3 *Description*

4 Converts the value of the specified double-precision floating point number
5 to an equivalent **Decimal** number.

6 *Return Value:* A **Decimal** number equivalent to the value of *value* . A double-
7 precision floating point number.

8 ToDecimal

10 [C#] public static decimal ToDecimal(short value);

11 [C++] public: static Decimal ToDecimal(short value);

12 [VB] Public Shared Function ToDecimal(ByVal value As Short) As Decimal

13 [JScript] public static function ToDecimal(value : Int16) : Decimal;

15 *Description*

16 Converts the value of the specified 16-bit signed integer to an equivalent
17 **Decimal** number.

18 *Return Value:* A **Decimal** number equivalent to the value of *value* . A 16-bit
19 signed integer.

20 ToDecimal

22 [C#] public static decimal ToDecimal(int value);

23 [C++] public: static Decimal ToDecimal(int value);

24 [VB] Public Shared Function ToDecimal(ByVal value As Integer) As Decimal

25 [JScript] public static function ToDecimal(value : int) : Decimal;

Description

Converts the value of the specified 32-bit signed integer to an equivalent

Decimal number.

Return Value: A **Decimal** number equivalent to the value of *value* . A 32-bit signed integer.

ToDecimal

[C#] public static decimal ToDecimal(long value);

[C++] public: static Decimal ToDecimal(__int64 value);

[VB] Public Shared Function ToDecimal(ByVal value As Long) As Decimal

[JScript] public static function ToDecimal(value : long) : Decimal;

Description

Converts the value of the specified 64-bit signed integer to an equivalent

Decimal number.

Return Value: A **Decimal** number equivalent to the value of *value* . A 64-bit signed integer.

ToDecimal

[C#] public static decimal ToDecimal(object value);

[C++] public: static Decimal ToDecimal(Object* value);

[VB] Public Shared Function ToDecimal(ByVal value As Object) As Decimal

[JScript] public static function ToDecimal(value : Object) : Decimal; Converts a specified value to a **Decimal** number.

1
2 *Description*

3 Converts the value of the specified **Object** to a **Decimal** number.

4 *Return Value:* A **Decimal** number equivalent to the value of *value* , or zero if
5 *value* is **null** .

6 The return value is the result of invoking the **IConvertible.ToDecimal**
7 method of the underlying type of *value* . An **System.Object** that implements the
8 **System.IConvertible** interface or **null**.

9 **ToDecimal**

10
11 [C#] public static decimal ToDecimal(sbyte value);

12 [C++] public: static Decimal ToDecimal(char value);

13 [VB] Public Shared Function ToDecimal(ByVal value As SByte) As Decimal

14 [JScript] public static function ToDecimal(value : SByte) : Decimal;

15
16 *Description*

17 Converts the value of the specified 8-bit signed integer to the equivalent
18 **Decimal** number.

19 *Return Value:* The 8-bit signed integer equivalent to the value of *value* . An 8-bit
20 signed integer.

21 **ToDecimal**

22
23 [C#] public static decimal ToDecimal(float value);

24 [C++] public: static Decimal ToDecimal(float value);

25 [VB] Public Shared Function ToDecimal(ByVal value As Single) As Decimal

1 [JScript] public static function ToDecimal(value : float) : Decimal;

3 *Description*

4 Converts the value of the specified single-precision floating point number
5 to the equivalent **Decimal** number.

6 *Return Value:* A **Decimal** number equivalent to the value of *value* . A single-
7 precision floating point number.

8 ToDecimal

10 [C#] public static decimal ToDecimal(string value);

11 [C++] public: static Decimal ToDecimal(String* value);

12 [VB] Public Shared Function ToDecimal(ByVal value As String) As Decimal

13 [JScript] public static function ToDecimal(value : String) : Decimal;

15 *Description*

16 Converts the specified **String** representation of a number to an equivalent
17 **Decimal** number.

18 *Return Value:* A **Decimal** number equivalent to the value of *value* .

19 The return value is the result of invoking the
20 **System.Decimal.Parse(System.String)** method on *value* . A **System.String**
21 containing a number to convert.

22 ToDecimal

24 [C#] public static decimal ToDecimal(ushort value);

25 [C++] public: static Decimal ToDecimal(unsigned short value);

1 [VB] Public Shared Function ToDecimal(ByVal value As UInt16) As Decimal

2 [JScript] public static function ToDecimal(value : UInt16) : Decimal;

3
4 *Description*

5 Converts the value of the specified 16-bit unsigned integer to the equivalent

6 **Decimal** number.

7 *Return Value:* The **Decimal** number equivalent to the value of *value* . A 16-bit
8 unsigned integer.

9 ToDecimal

10
11 [C#] public static decimal ToDecimal(uint value);

12 [C++] public: static Decimal ToDecimal(unsigned int value);

13 [VB] Public Shared Function ToDecimal(ByVal value As UInt32) As Decimal

14 [JScript] public static function ToDecimal(value : UInt32) : Decimal;

15
16 *Description*

17 Converts the value of the specified 32-bit unsigned integer to an equivalent

18 **Decimal** number.

19 *Return Value:* A **Decimal** number equivalent to the value of *value* . A 32-bit
20 unsigned integer.

21 ToDecimal

22
23 [C#] public static decimal ToDecimal(ulong value);

24 [C++] public: static Decimal ToDecimal(unsigned __int64 value);

25 [VB] Public Shared Function ToDecimal(ByVal value As UInt64) As Decimal

1 [JScript] public static function ToDecimal(value : UInt64) : Decimal;

3 *Description*

4 Converts the value of the specified 64-bit unsigned integer to an equivalent

5 **Decimal** number.

6 *Return Value:* A **Decimal** number equivalent to the value of *value* . A 64-bit
7 unsigned integer.

8 ToDecimal

10 [C#] public static decimal ToDecimal(object value, IFormatProvider provider);

11 [C++] public: static Decimal ToDecimal(Object* value, IFormatProvider*
12 provider);

13 [VB] Public Shared Function ToDecimal(ByVal value As Object, ByVal provider
14 As IFormatProvider) As Decimal

15 [JScript] public static function ToDecimal(value : Object, provider :
16 IFormatProvider) : Decimal;

18 *Description*

19 Converts the value of the specified **Object** to an **Decimal** number using the
20 specified culture-specific formatting information.

21 *Return Value:* A **Decimal** number equivalent to the value of *value* , or zero if
22 *value* is **null** .

23 The return value is the result of invoking the **IConvertible.ToDecimal**
24 method of the underlying type of *value* . An **System.Object** that implements the

System.IConvertible interface. An **System.IFormatProvider** interface implementation that supplies culture-specific formatting information.

ToDecimal

[C#] public static decimal ToDecimal(string value, IFormatProvider provider);

[C++] public: static Decimal ToDecimal(String* value, IFormatProvider* provider);

[VB] Public Shared Function ToDecimal(ByVal value As String, ByVal provider As IFormatProvider) As Decimal

[JScript] public static function ToDecimal(value : String, provider : IFormatProvider) : Decimal;

Description

Converts the specified **String** representation of a number to an equivalent **Decimal** number using the specified culture-specific formatting information.

Return Value: A **Decimal** number equivalent to the value of *value* .

The return value is the result of invoking the **System.Decimal.Parse(System.String)** method on *value* . A **System.String** containing a number to convert. An **System.IFormatProvider** interface implementation that supplies culture-specific formatting information.

ToDouble

[C#] public static double ToDouble(bool value);

[C++] public: static double ToDouble(bool value);

[VB] Public Shared Function ToDouble(ByVal value As Boolean) As Double

1 [JScript] public static function ToDouble(value : Boolean) : double;

3 *Description*

4 Converts the value of the specified Boolean value to the equivalent double-
5 precision floating point number.

6 *Return Value:* The number 1 if *value* is **true** ; otherwise, 0. A Boolean value.

7 ToDouble

9 [C#] public static double ToDouble(byte value);

10 [C++] public: static double ToDouble(unsigned char value);

11 [VB] Public Shared Function ToDouble(ByVal value As Byte) As Double

12 [JScript] public static function ToDouble(value : Byte) : double;

14 *Description*

15 Converts the value of the specified 8-bit unsigned integer to the equivalent
16 double-precision floating point number.

17 *Return Value:* The double-precision floating point number equivalent to the value
18 of *value* . An 8-bit unsigned integer.

19 ToDouble

21 [C#] public static double ToDouble(char value);

22 [C++] public: static double ToDouble(__wchar_t value);

23 [VB] Public Shared Function ToDouble(ByVal value As Char) As Double

24 [JScript] public static function ToDouble(value : Char) : double;

Description

Calling this method always throws **System.InvalidCastException** .

This method is reserved for future use. A Unicode character.

ToDouble

[C#] public static double ToDouble(DateTime value);

[C++] public: static double ToDouble(DateTime value);

[VB] Public Shared Function ToDouble(ByVal value As DateTime) As Double

[JScript] public static function ToDouble(value : DateTime) : double;

Description

Calling this method always throws **System.InvalidCastException** .

This method is reserved for future use. A **System.DateTime**.

ToDouble

[C#] public static double ToDouble(decimal value);

[C++] public: static double ToDouble(Decimal value);

[VB] Public Shared Function ToDouble(ByVal value As Decimal) As Double

[JScript] public static function ToDouble(value : Decimal) : double;

Description

Converts the value of the specified **Decimal** number to an equivalent double-precision floating point number.

Return Value: A double-precision floating point number equivalent to the value of *value* . A **System.Decimal** number.

ToDouble

[C#] public static double ToDouble(double value);

[C++] public: static double ToDouble(double value);

[VB] Public Shared Function ToDouble(ByVal value As Double) As Double

[JScript] public static function ToDouble(value : double) : double;

Description

Returns the specified double-precision floating point number; no actual conversion is performed.

Return Value: *value* is returned unchanged. A double-precision floating point number.

ToDouble

[C#] public static double ToDouble(short value);

[C++] public: static double ToDouble(short value);

[VB] Public Shared Function ToDouble(ByVal value As Short) As Double

[JScript] public static function ToDouble(value : Int16) : double;

Description

Converts the value of the specified 16-bit signed integer to an equivalent double-precision floating point number.

Return Value: A double-precision floating point number equivalent to the value of *value* . A 16-bit signed integer.

ToDouble

[C#] public static double ToDouble(int value);

[C++] public: static double ToDouble(int value);

[VB] Public Shared Function ToDouble(ByVal value As Integer) As Double

[JScript] public static function ToDouble(value : int) : double;

Description

Converts the value of the specified 32-bit signed integer to an equivalent double-precision floating point number.

Return Value: A double-precision floating point number equivalent to the value of *value* . A 32-bit signed integer.

ToDouble

[C#] public static double ToDouble(long value);

[C++] public: static double ToDouble(__int64 value);

[VB] Public Shared Function ToDouble(ByVal value As Long) As Double

[JScript] public static function ToDouble(value : long) : double;

Description

Converts the value of the specified 64-bit signed integer to an equivalent double-precision floating point number.

Return Value: A double-precision floating point number equivalent to the value of *value* . A 64-bit signed integer.

ToDouble

[C#] public static double ToDouble(object value);

[C++] public: static double ToDouble(Object* value);

[VB] Public Shared Function ToDouble(ByVal value As Object) As Double

[JScript] public static function ToDouble(value : Object) : double; Converts a specified value to a double-precision floating point number.

Description

Converts the value of the specified **Object** to a double-precision floating point number.

Return Value: A double-precision floating point number equivalent to the value of *value* , or zero if *value* is **null** .

The return value is the result of invoking the **IConvertible.ToDouble** method of the underlying type of *value* . An **System.Object** that implements the **System.IConvertible** interface or **null**.

ToDouble

[C#] public static double ToDouble(sbyte value);

[C++] public: static double ToDouble(char value);

[VB] Public Shared Function ToDouble(ByVal value As SByte) As Double

[JScript] public static function ToDouble(value : SByte) : double;

1
2 *Description*

3 Converts the value of the specified 8-bit signed integer to the equivalent
4 double-precision floating point number.

5 *Return Value:* The 8-bit signed integer equivalent to the value of *value* . An 8-bit
6 signed integer.

7 ToDouble

8
9 [C#] public static double ToDouble(float value);

10 [C++] public: static double ToDouble(float value);

11 [VB] Public Shared Function ToDouble(ByVal value As Single) As Double

12 [JScript] public static function ToDouble(value : float) : double;

13
14 *Description*

15 Converts the value of the specified single-precision floating point number
16 to an equivalent double-precision floating point number.

17 *Return Value:* A double-precision floating point number equivalent to the value of
18 *value* . A single-precision floating point number.

19 ToDouble

20
21 [C#] public static double ToDouble(string value);

22 [C++] public: static double ToDouble(String* value);

23 [VB] Public Shared Function ToDouble(ByVal value As String) As Double

24 [JScript] public static function ToDouble(value : String) : double;

Description

Converts the specified **String** representation of a number to an equivalent double-precision floating point number.

Return Value: A double-precision floating point number equivalent to the value of *value*.

The return value is the result of invoking the **System.Double.Parse(System.String)** method on *value*. A **System.String** containing a number to convert.

ToDouble

[C#] public static double ToDouble(ushort value);

[C++] public: static double ToDouble(unsigned short value);

[VB] Public Shared Function ToDouble(ByVal value As UInt16) As Double

[JScript] public static function ToDouble(value : UInt16) : double;

Description

Converts the value of the specified 16-bit unsigned integer to the equivalent double-precision floating point number.

Return Value: The double-precision floating point number equivalent to the value of *value*. A 16-bit unsigned integer.

ToDouble

[C#] public static double ToDouble(uint value);

[C++] public: static double ToDouble(unsigned int value);

1 [VB] Public Shared Function ToDouble(ByVal value As UInt32) As Double

2 [JScript] public static function ToDouble(value : UInt32) : double;

3
4 *Description*

5 Converts the value of the specified 32-bit unsigned integer to an equivalent
6 double-precision floating point number.

7 *Return Value:* A double-precision floating point number equivalent to the value of
8 *value* . A 32-bit unsigned integer.

9 ToDouble

10
11 [C#] public static double ToDouble(ulong value);

12 [C++] public: static double ToDouble(unsigned __int64 value);

13 [VB] Public Shared Function ToDouble(ByVal value As UInt64) As Double

14 [JScript] public static function ToDouble(value : UInt64) : double;

15
16 *Description*

17 Converts the value of the specified 64-bit unsigned integer to an equivalent
18 double-precision floating point number.

19 *Return Value:* A double-precision floating point number equivalent to the value of
20 *value* . A 64-bit unsigned integer.

21 ToDouble

22
23 [C#] public static double ToDouble(object value, IFormatProvider provider);

24 [C++] public: static double ToDouble(Object* value, IFormatProvider* provider);

25 [VB] Public Shared Function ToDouble(ByVal value As Object, ByVal provider

As IFormatProvider) As Double

[JScript] public static function ToDouble(value : Object, provider :
IFormatProvider) : double;

Description

Converts the value of the specified **Object** to an double-precision floating point number using the specified culture-specific formatting information.

Return Value: A double-precision floating point number equivalent to the value of *value* , or zero if *value* is **null** .

The return value is the result of invoking the **IConvertible.ToDouble** method of the underlying type of *value* . An **System.Object** that implements the **System.IConvertible** interface. An **System.IFormatProvider** interface implementation that supplies culture-specific formatting information.

ToDouble

[C#] public static double ToDouble(string value, IFormatProvider provider);

[C++] public: static double ToDouble(String* value, IFormatProvider* provider);

[VB] Public Shared Function ToDouble(ByVal value As String, ByVal provider

As IFormatProvider) As Double

[JScript] public static function ToDouble(value : String, provider :
IFormatProvider) : double;

Description

Converts the specified **String** representation of a number to an equivalent double-precision floating point number using the specified culture-specific

1 formatting information.

2 *Return Value:* A double-precision floating point number equivalent to the value of
3 *value* .

4 The return value is the result of invoking the
5 **System.Double.Parse(System.String)** method on *value* . A **System.String**
6 containing a number to convert. An **System.IFormatProvider** interface
7 implementation that supplies culture-specific formatting information.

8 **ToInt16**

9
10 [C#] public static short ToInt16(bool value);

11 [C++] public: static short ToInt16(bool value);

12 [VB] Public Shared Function ToInt16(ByVal value As Boolean) As Short

13 [JScript] public static function ToInt16(value : Boolean) : Int16;

14
15 *Description*

16 Converts the value of the specified Boolean value to the equivalent 16-bit
17 signed integer.

18 *Return Value:* The number 1 if *value* is **true** ; otherwise, 0. A Boolean value.

19 **ToInt16**

20
21 [C#] public static short ToInt16(byte value);

22 [C++] public: static short ToInt16(unsigned char value);

23 [VB] Public Shared Function ToInt16(ByVal value As Byte) As Short

24 [JScript] public static function ToInt16(value : Byte) : Int16;

1
2 *Description*

3 Converts the value of the specified 8-bit unsigned integer to the equivalent
4 16-bit signed integer.

5 *Return Value:* The 16-bit signed integer equivalent to the value of *value* . An 8-bit
6 unsigned integer.

7 ToInt16

8
9 [C#] public static short ToInt16(char value);

10 [C++] public: static short ToInt16(__wchar_t value);

11 [VB] Public Shared Function ToInt16(ByVal value As Char) As Short

12 [JScript] public static function ToInt16(value : Char) : Int16;

13
14 *Description*

15 Converts the value of the specified Unicode character to the equivalent 16-
16 bit signed integer.

17 *Return Value:* The 16-bit signed integer equivalent to *value* . A Unicode character.

18 ToInt16

19
20 [C#] public static short ToInt16(DateTime value);

21 [C++] public: static short ToInt16(DateTime value);

22 [VB] Public Shared Function ToInt16(ByVal value As DateTime) As Short

23 [JScript] public static function ToInt16(value : DateTime) : Int16;

24
25 *Description*

Calling this method always throws **System.InvalidCastException** .

This method is reserved for future use. A **System.DateTime**.

ToInt16

[C#] public static short ToInt16(decimal value);

[C++] public: static short ToInt16(Decimal value);

[VB] Public Shared Function ToInt16(ByVal value As Decimal) As Short

[JScript] public static function ToInt16(value : Decimal) : Int16;

Description

Converts the value of the specified **Decimal** number to an equivalent 16-bit signed integer.

Return Value: *value* rounded to the nearest 16-bit signed integer. If *value* is halfway between two whole numbers, the even number is returned; that is, 4.5 is converted to 4, and 5.5 is converted to 6. A **System.Decimal** number.

ToInt16

[C#] public static short ToInt16(double value);

[C++] public: static short ToInt16(double value);

[VB] Public Shared Function ToInt16(ByVal value As Double) As Short

[JScript] public static function ToInt16(value : double) : Int16;

Description

Converts the value of the specified double-precision floating point number to an equivalent 16-bit signed integer.

Return Value: *value* rounded to the nearest 16-bit signed integer. If *value* is halfway between two whole numbers, the even number is returned; that is, 4.5 is converted to 4, and 5.5 is converted to 6. A double-precision floating point number.

ToInt16

[C#] public static short ToInt16(short value);

[C++] public: static short ToInt16(short value);

[VB] Public Shared Function ToInt16(ByVal value As Short) As Short

[JScript] public static function ToInt16(value : Int16) : Int16;

Description

Returns the specified 16-bit signed integer; no actual conversion is performed.

Return Value: *value* is returned unchanged. A 16-bit signed integer.

ToInt16

[C#] public static short ToInt16(int value);

[C++] public: static short ToInt16(int value);

[VB] Public Shared Function ToInt16(ByVal value As Integer) As Short

[JScript] public static function ToInt16(value : int) : Int16;

Description

Converts the value of the specified 32-bit signed integer to an equivalent 16-bit signed integer.

Return Value: The 16-bit signed integer equivalent of *value* . A 32-bit signed integer.

ToInt16

[C#] public static short ToInt16(long value);

[C++] public: static short ToInt16(__int64 value);

[VB] Public Shared Function ToInt16(ByVal value As Long) As Short

[JScript] public static function ToInt16(value : long) : Int16;

Description

Converts the value of the specified 64-bit signed integer to an equivalent 16-bit signed integer.

Return Value: A 16-bit signed integer equivalent to the value of *value* . A 64-bit signed integer.

ToInt16

[C#] public static short ToInt16(object value);

[C++] public: static short ToInt16(Object* value);

[VB] Public Shared Function ToInt16(ByVal value As Object) As Short

[JScript] public static function ToInt16(value : Object) : Int16; Converts a specified value to a 16-bit signed integer.

Description

Converts the value of the specified **Object** to a 16-bit signed integer.

Return Value: A 16-bit signed integer equivalent to the value of *value* , or zero if *value* is **null** .

The return value is the result of invoking the **IConvertible.ToInt16** method of the underlying type of *value* . An **System.Object** that implements the **System.IConvertible** interface or **null**.

ToInt16

[C#] public static short ToInt16(sbyte value);

[C++] public: static short ToInt16(char value);

[VB] Public Shared Function ToInt16(ByVal value As SByte) As Short

[JScript] public static function ToInt16(value : SByte) : Int16;

Description

Converts the value of the specified 8-bit signed integer to the equivalent 16-bit signed integer.

Return Value: The 8-bit signed integer equivalent to the value of *value* . An 8-bit signed integer.

ToInt16

[C#] public static short ToInt16(float value);

[C++] public: static short ToInt16(float value);

[VB] Public Shared Function ToInt16(ByVal value As Single) As Short

[JScript] public static function ToInt16(value : float) : Int16;

1
2 *Description*

3 Converts the value of the specified single-precision floating point number
4 to an equivalent 16-bit signed integer.

5 *Return Value:* *value* rounded to the nearest 16-bit signed integer. If *value* is
6 halfway between two whole numbers, the even number is returned; that is, 4.5 is
7 converted to 4, and 5.5 is converted to 6. A single-precision floating point number.

8 ToInt16

9
10 [C#] public static short ToInt16(string value);

11 [C++] public: static short ToInt16(String* value);

12 [VB] Public Shared Function ToInt16(ByVal value As String) As Short

13 [JScript] public static function ToInt16(value : String) : Int16;

14
15 *Description*

16 Converts the specified **String** representation of a number to an equivalent
17 16-bit signed integer.

18 *Return Value:* A 16-bit signed integer equivalent to the value of *value* . A

19 **System.String** containing a number to convert.

20 ToInt16

21
22 [C#] public static short ToInt16(ushort value);

23 [C++] public: static short ToInt16(unsigned short value);

24 [VB] Public Shared Function ToInt16(ByVal value As UInt16) As Short

25 [JScript] public static function ToInt16(value : UInt16) : Int16;

1
2 *Description*

3 Converts the value of the specified 16-bit unsigned integer to the equivalent
4 16-bit signed integer.

5 *Return Value:* The 16-bit signed integer equivalent to the value of *value* . A 16-bit
6 unsigned integer.

7 ToInt16

8
9 [C#] public static short ToInt16(uint value);

10 [C++] public: static short ToInt16(unsigned int value);

11 [VB] Public Shared Function ToInt16(ByVal value As UInt32) As Short

12 [JScript] public static function ToInt16(value : UInt32) : Int16;

13
14 *Description*

15 Converts the value of the specified 32-bit unsigned integer to an equivalent
16 16-bit signed integer.

17 *Return Value:* A 16-bit signed integer equivalent to the value of *value* . A 32-bit
18 unsigned integer.

19 ToInt16

20
21 [C#] public static short ToInt16(ulong value);

22 [C++] public: static short ToInt16(unsigned __int64 value);

23 [VB] Public Shared Function ToInt16(ByVal value As UInt64) As Short

24 [JScript] public static function ToInt16(value : UInt64) : Int16;

1
2 *Description*

3 Converts the value of the specified 64-bit unsigned integer to an equivalent
4 16-bit signed integer.

5 *Return Value:* A 16-bit signed integer equivalent to the value of *value* . A 64-bit
6 unsigned integer.

7 ToInt16

8
9 [C#] public static short ToInt16(object value, IFormatProvider provider);

10 [C++] public: static short ToInt16(Object* value, IFormatProvider* provider);

11 [VB] Public Shared Function ToInt16(ByVal value As Object, ByVal provider As
12 IFormatProvider) As Short

13 [JScript] public static function ToInt16(value : Object, provider : IFormatProvider)
14 : Int16;

15
16 *Description*

17 Converts the value of the specified **Object** to a 16-bit signed integer using
18 the specified culture-specific formatting information.

19 *Return Value:* A 16-bit signed integer equivalent to the value of *value* , or zero if
20 *value* is **null** .

21 *provider* enables the user to specify culture-specific conversion information
22 about the contents of *value* . For example, if *value* is a **String** that represents a
23 number, *provider* could supply culture-specific information about the notation
24 used to represent that number. An **System.Object** that implements the

System.IConvertible interface. An **System.IFormatProvider** interface implementation that supplies culture-specific formatting information.

ToInt16

[C#] public static short ToInt16(string value, IFormatProvider provider);

[C++] public: static short ToInt16(String* value, IFormatProvider* provider);

[VB] Public Shared Function ToInt16(ByVal value As String, ByVal provider As IFormatProvider) As Short

[JScript] public static function ToInt16(value : String, provider : IFormatProvider) : Int16;

Description

Converts the specified **String** representation of a number to an equivalent 16-bit signed integer using specified culture-specific formatting information.

Return Value: A 16-bit signed integer equivalent to the value of *value* .

provider is an **IFormatProvider** instance that obtains a **System.Globalization.NumberFormatInfo** object. The **NumberFormatInfo** object provides culture-specific information about the format of *value* . If *provider* is **null** , the **NumberFormatInfo** for the current culture is used. A **System.String** containing a number to convert. An **System.IFormatProvider** interface implementation that supplies culture-specific formatting information.

ToInt16

[C#] public static short ToInt16(string value, int fromBase);

[C++] public: static short ToInt16(String* value, int fromBase);

1 [VB] Public Shared Function ToInt16(ByVal value As String, ByVal fromBase As
2 Integer) As Short

3 [JScript] public static function ToInt16(value : String, fromBase : int) : Int16;

4
5 *Description*

6 Converts the **String** representation of a number in a specified base to an
7 equivalent 16-bit signed integer.

8 *Return Value:* A 16-bit signed integer equivalent to the number in *value* . A

9 **System.String** containing a number. The base of the number in *value*, which must
10 be 2, 8, 10, or 16.

11 ToInt32

12
13 [C#] public static int ToInt32(bool value);

14 [C++] public: static int ToInt32(bool value);

15 [VB] Public Shared Function ToInt32(ByVal value As Boolean) As Integer

16 [JScript] public static function ToInt32(value : Boolean) : int;

17
18 *Description*

19 Converts the value of the specified Boolean value to the equivalent 32-bit
20 signed integer.

21 *Return Value:* The number 1 if *value* is **true** ; otherwise, 0. A Boolean value.

22 ToInt32

23
24 [C#] public static int ToInt32(byte value);

25 [C++] public: static int ToInt32(unsigned char value);

[VB] Public Shared Function ToInt32(Byte value As Byte) As Integer

[JScript] public static function ToInt32(value : Byte) : int;

Description

Converts the value of the specified 8-bit unsigned integer to the equivalent 32-bit signed integer.

Return Value: The 32-bit signed integer equivalent to the value of *value* . An 8-bit unsigned integer.

ToInt32

[C#] public static int ToInt32(char value);

[C++] public: static int ToInt32(__wchar_t value);

[VB] Public Shared Function ToInt32(Byte value As Char) As Integer

[JScript] public static function ToInt32(value : Char) : int;

Description

Converts the value of the specified Unicode character to the equivalent 32-bit signed integer.

Return Value: The 32-bit signed integer equivalent to *value* . A Unicode character.

ToInt32

[C#] public static int ToInt32(DateTime value);

[C++] public: static int ToInt32(DateTime value);

[VB] Public Shared Function ToInt32(Byte value As DateTime) As Integer

[JScript] public static function ToInt32(value : DateTime) : int;

1
2 *Description*

3 Calling this method always throws **System.InvalidCastException** .

4 This method is reserved for future use. A **System.DateTime**.

5 ToInt32

6
7 [C#] public static int ToInt32(decimal value);

8 [C++] public: static int ToInt32(Decimal value);

9 [VB] Public Shared Function ToInt32(ByVal value As Decimal) As Integer

10 [JScript] public static function ToInt32(value : Decimal) : int;

11
12 *Description*

13 Converts the value of the specified **Decimal** number to an equivalent 32-bit
14 signed integer.

15 *Return Value:* *value* rounded to the nearest 32-bit signed integer. If *value* is
16 halfway between two whole numbers, the even number is returned; that is, 4.5 is
17 converted to 4, and 5.5 is converted to 6. A **System.Decimal** number.

18 ToInt32

19
20 [C#] public static int ToInt32(double value);

21 [C++] public: static int ToInt32(double value);

22 [VB] Public Shared Function ToInt32(ByVal value As Double) As Integer

23 [JScript] public static function ToInt32(value : double) : int;

24
25 *Description*

Converts the value of the specified double-precision floating point number to an equivalent 32-bit signed integer.

Return Value: *value* rounded to the nearest 32-bit signed integer. If *value* is halfway between two whole numbers, the even number is returned; that is, 4.5 is converted to 4, and 5.5 is converted to 6. A double-precision floating point number.

ToInt32

[C#] public static int ToInt32(short value);

[C++] public: static int ToInt32(short value);

[VB] Public Shared Function ToInt32(ByVal value As Short) As Integer

[JScript] public static function ToInt32(value : Int16) : int;

Description

Converts the value of the specified 16-bit signed integer to an equivalent 32-bit signed integer.

Return Value: A 32-bit signed integer equivalent to the value of *value* . A 16-bit signed integer.

ToInt32

[C#] public static int ToInt32(int value);

[C++] public: static int ToInt32(int value);

[VB] Public Shared Function ToInt32(ByVal value As Integer) As Integer

[JScript] public static function ToInt32(value : int) : int;

1
2 *Description*

3 Returns the specified 32-bit signed integer; no actual conversion is
4 performed.

5 *Return Value:* *value* is returned unchanged. A 32-bit signed integer.

6 **ToInt32**

7
8 [C#] public static int ToInt32(long value);

9 [C++] public: static int ToInt32(__int64 value);

10 [VB] Public Shared Function ToInt32(ByVal value As Long) As Integer

11 [JScript] public static function ToInt32(value : long) : int;

12
13 *Description*

14 Converts the value of the specified 64-bit signed integer to an equivalent
15 32-bit signed integer.

16 *Return Value:* A 32-bit signed integer equivalent to the value of *value* . A 64-bit
17 signed integer.

18 **ToInt32**

19
20 [C#] public static int ToInt32(object value);

21 [C++] public: static int ToInt32(Object* value);

22 [VB] Public Shared Function ToInt32(ByVal value As Object) As Integer

23 [JScript] public static function ToInt32(value : Object) : int; Converts a specified
24 value to a 32-bit signed integer.
25

Description

Converts the value of the specified **Object** to a 32-bit signed integer.

Return Value: A 32-bit signed integer equivalent to the value of *value* , or zero if *value* is **null** .

The return value is the result of invoking the **ICollection.ToInt32** method of the underlying type of *value* . An **System.Object** that implements the **System.ICollection** interface or **null**.

ToInt32

[C#] public static int ToInt32(sbyte value);

[C++] public: static int ToInt32(char value);

[VB] Public Shared Function ToInt32(ByVal value As SByte) As Integer

[JScript] public static function ToInt32(value : SByte) : int;

Description

Converts the value of the specified 8-bit signed integer to the equivalent 32-bit signed integer.

Return Value: The 8-bit signed integer equivalent to the value of *value* . An 8-bit signed integer.

ToInt32

[C#] public static int ToInt32(float value);

[C++] public: static int ToInt32(float value);

[VB] Public Shared Function ToInt32(ByVal value As Single) As Integer

1 [JScript] public static function ToInt32(value : float) : int;

3 *Description*

4 Converts the value of the specified single-precision floating point number
5 to an equivalent 32-bit signed integer.

6 *Return Value:* *value* rounded to the nearest 32-bit signed integer. If *value* is
7 halfway between two whole numbers, the even number is returned; that is, 4.5 is
8 converted to 4, and 5.5 is converted to 6. A single-precision floating point number.

9 ToInt32

11 [C#] public static int ToInt32(string value);

12 [C++] public: static int ToInt32(String* value);

13 [VB] Public Shared Function ToInt32(ByVal value As String) As Integer

14 [JScript] public static function ToInt32(value : String) : int;

16 *Description*

17 Converts the specified **String** representation of a number to an equivalent
18 32-bit signed integer.

19 *Return Value:* A 32-bit signed integer equivalent to the value of *value* .

20 The return value is the result of invoking the
21 **System.Int32.Parse(System.String)** method on *value* . A **System.String**
22 containing a number to convert.

23 ToInt32

25 [C#] public static int ToInt32(ushort value);

[C++] public: static int ToInt32(unsigned short value);

[VB] Public Shared Function ToInt32(ByVal value As UInt16) As Integer

[JScript] public static function ToInt32(value : UInt16) : int;

Description

Converts the value of the specified 16-bit unsigned integer to the equivalent 32-bit signed integer.

Return Value: The 32-bit signed integer equivalent to the value of *value* . A 16-bit unsigned integer.

ToInt32

[C#] public static int ToInt32(uint value);

[C++] public: static int ToInt32(unsigned int value);

[VB] Public Shared Function ToInt32(ByVal value As UInt32) As Integer

[JScript] public static function ToInt32(value : UInt32) : int;

Description

Converts the value of the specified 32-bit unsigned integer to an equivalent 32-bit signed integer.

Return Value: A 32-bit signed integer equivalent to the value of *value* . A 32-bit unsigned integer.

ToInt32

[C#] public static int ToInt32(ulong value);

[C++] public: static int ToInt32(unsigned __int64 value);

1 [VB] Public Shared Function ToInt32(ByVal value As UInt64) As Integer

2 [JScript] public static function ToInt32(value : UInt64) : int;

3
4 *Description*

5 Converts the value of the specified 64-bit unsigned integer to an equivalent
6 32-bit signed integer.

7 *Return Value:* A 32-bit signed integer equivalent to the value of *value* . A 64-bit
8 unsigned integer.

9 ToInt32

10
11 [C#] public static int ToInt32(object value, IFormatProvider provider);

12 [C++] public: static int ToInt32(Object* value, IFormatProvider* provider);

13 [VB] Public Shared Function ToInt32(ByVal value As Object, ByVal provider As
14 IFormatProvider) As Integer

15 [JScript] public static function ToInt32(value : Object, provider : IFormatProvider)
16 : int;

17
18 *Description*

19 Converts the value of the specified **Object** to a 32-bit signed integer using
20 the specified culture-specific formatting information.

21 *Return Value:* A 32-bit signed integer equivalent to the value of *value* , or zero if
22 *value* is **null** .

23 The return value is the result of invoking the **IConvertible.ToInt32**
24 method of the underlying type of *value* . An **System.Object** that implements the

System.IConvertible interface. An **System.IFormatProvider** interface implementation that supplies culture-specific formatting information.

ToInt32

[C#] public static int ToInt32(string value, IFormatProvider provider);

[C++] public: static int ToInt32(String* value, IFormatProvider* provider);

[VB] Public Shared Function ToInt32(ByVal value As String, ByVal provider As IFormatProvider) As Integer

[JScript] public static function ToInt32(value : String, provider : IFormatProvider) : int;

Description

Converts the specified **String** representation of a number to an equivalent 32-bit signed integer using specified culture-specific formatting information.

Return Value: A 32-bit signed integer equivalent to the value of *value* .

The return value is the result of invoking the **System.Int32.Parse(System.String)** method on *value* . A **System.String** containing a number to convert. An **System.IFormatProvider** interface implementation that supplies culture-specific formatting information.

ToInt32

[C#] public static int ToInt32(string value, int fromBase);

[C++] public: static int ToInt32(String* value, int fromBase);

[VB] Public Shared Function ToInt32(ByVal value As String, ByVal fromBase As Integer) As Integer

1 [JScript] public static function ToInt32(value : String, fromBase : int) : int;

3 *Description*

4 Converts the **String** representation of a number in a specified base to an
5 equivalent 32-bit signed integer.

6 *Return Value:* A 32-bit signed integer equivalent to the number in *value* . A

7 **System.String** containing a number. The base of the number in *value*, which must
8 be 2, 8, 10, or 16.

9 **ToInt64**

10
11 [C#] public static long ToInt64(bool value);

12 [C++] public: static __int64 ToInt64(bool value);

13 [VB] Public Shared Function ToInt64(ByVal value As Boolean) As Long

14 [JScript] public static function ToInt64(value : Boolean) : long;

15
16 *Description*

17 Converts the value of the specified Boolean value to the equivalent 64-bit
18 signed integer.

19 *Return Value:* The number 1 if *value* is **true** ; otherwise, 0. A Boolean value.

20 **ToInt64**

21
22 [C#] public static long ToInt64(byte value);

23 [C++] public: static __int64 ToInt64(unsigned char value);

24 [VB] Public Shared Function ToInt64(ByVal value As Byte) As Long

25 [JScript] public static function ToInt64(value : Byte) : long;

1
2 *Description*

3 Converts the value of the specified 8-bit unsigned integer to the equivalent
4 64-bit signed integer.

5 *Return Value:* The 64-bit signed integer equivalent to the value of *value* . An 8-bit
6 unsigned integer.

7 ToInt64

8
9 [C#] public static long ToInt64(char value);

10 [C++] public: static __int64 ToInt64(__wchar_t value);

11 [VB] Public Shared Function ToInt64(ByVal value As Char) As Long

12 [JScript] public static function ToInt64(value : Char) : long;

13
14 *Description*

15 Converts the value of the specified Unicode character to the equivalent 64-
16 bit signed integer.

17 *Return Value:* The 64-bit signed integer equivalent to *value* . A Unicode character.

18 ToInt64

19
20 [C#] public static long ToInt64(DateTime value);

21 [C++] public: static __int64 ToInt64(DateTime value);

22 [VB] Public Shared Function ToInt64(ByVal value As DateTime) As Long

23 [JScript] public static function ToInt64(value : DateTime) : long;

24
25 *Description*

Calling this method always throws **System.InvalidCastException** .

This method is reserved for future use. A **System.DateTime**.

ToInt64

[C#] public static long ToInt64(decimal value);

[C++] public: static __int64 ToInt64(Decimal value);

[VB] Public Shared Function ToInt64(ByVal value As Decimal) As Long

[JScript] public static function ToInt64(value : Decimal) : long;

Description

Converts the value of the specified **Decimal** number to an equivalent 64-bit signed integer.

Return Value: *value* rounded to the nearest 64-bit signed integer. If *value* is halfway between two whole numbers, the even number is returned; that is, 4.5 is converted to 4, and 5.5 is converted to 6. A **System.Decimal** number.

ToInt64

[C#] public static long ToInt64(double value);

[C++] public: static __int64 ToInt64(double value);

[VB] Public Shared Function ToInt64(ByVal value As Double) As Long

[JScript] public static function ToInt64(value : double) : long;

Description

Converts the value of the specified double-precision floating point number to an equivalent 64-bit signed integer.

Return Value: *value* rounded to the nearest 64-bit signed integer. If *value* is halfway between two whole numbers, the even number is returned; that is, 4.5 is converted to 4, and 5.5 is converted to 6. A double-precision floating point number.

ToInt64

[C#] public static long ToInt64(short value);

[C++] public: static __int64 ToInt64(short value);

[VB] Public Shared Function ToInt64(ByVal value As Short) As Long

[JScript] public static function ToInt64(value : Int16) : long;

Description

Converts the value of the specified 16-bit signed integer to an equivalent 64-bit signed integer.

Return Value: A 64-bit signed integer equivalent to the value of *value* . A 16-bit signed integer.

ToInt64

[C#] public static long ToInt64(int value);

[C++] public: static __int64 ToInt64(int value);

[VB] Public Shared Function ToInt64(ByVal value As Integer) As Long

[JScript] public static function ToInt64(value : int) : long;

Description

Converts the value of the specified 32-bit signed integer to an equivalent 64-bit signed integer.

Return Value: The 64-bit signed integer equivalent to the value of *value* . A 32-bit signed integer.

ToInt64

[C#] public static long ToInt64(long value);

[C++] public: static __int64 ToInt64(__int64 value);

[VB] Public Shared Function ToInt64(ByVal value As Long) As Long

[JScript] public static function ToInt64(value : long) : long;

Description

Returns the specified 64-bit signed integer; no actual conversion is performed.

Return Value: *value* is returned unchanged. A 64-bit signed integer.

ToInt64

[C#] public static long ToInt64(object value);

[C++] public: static __int64 ToInt64(Object* value);

[VB] Public Shared Function ToInt64(ByVal value As Object) As Long

[JScript] public static function ToInt64(value : Object) : long; Converts a specified value to a 64-bit signed integer.

Description

Converts the value of the specified **Object** to a 64-bit signed integer.

Return Value: A 64-bit signed integer equivalent to the value of *value* , or zero if *value* is **null** .

The return value is the result of invoking the **ICollection.ToInt64** method of the underlying type of *value* . An **System.Object** that implements the **System.ICollection** interface or **null**.

ToInt64

[C#] public static long ToInt64(sbyte value);

[C++] public: static __int64 ToInt64(char value);

[VB] Public Shared Function ToInt64(ByVal value As SByte) As Long

[JScript] public static function ToInt64(value : SByte) : long;

Description

Converts the value of the specified 8-bit signed integer to the equivalent 64-bit signed integer.

Return Value: The 8-bit signed integer equivalent to the value of *value* . An 8-bit signed integer.

ToInt64

[C#] public static long ToInt64(float value);

[C++] public: static __int64 ToInt64(float value);

[VB] Public Shared Function ToInt64(ByVal value As Single) As Long

[JScript] public static function ToInt64(value : float) : long;

1
2 *Description*

3 Converts the value of the specified single-precision floating point number
4 to an equivalent 64-bit signed integer.

5 *Return Value:* *value* rounded to the nearest 64-bit signed integer. If *value* is
6 halfway between two whole numbers, the even number is returned; that is, 4.5 is
7 converted to 4, and 5.5 is converted to 6. A single-precision floating point number.

8 **ToInt64**

9
10 [C#] public static long ToInt64(string value);

11 [C++] public: static __int64 ToInt64(String* value);

12 [VB] Public Shared Function ToInt64(ByVal value As String) As Long

13 [JScript] public static function ToInt64(value : String) : long;

14
15 *Description*

16 Converts the specified **String** representation of a number to an equivalent
17 64-bit signed integer.

18 *Return Value:* A 64-bit signed integer equivalent to the value of *value* .

19 The return value is the result of invoking the
20 **System.Int64.Parse(System.String)** method on *value* . A **System.String**
21 containing a number to convert.

22 **ToInt64**

23
24 [C#] public static long ToInt64(ushort value);

25 [C++] public: static __int64 ToInt64(unsigned short value);

1 [VB] Public Shared Function ToInt64(ByVal value As UInt16) As Long

2 [JScript] public static function ToInt64(value : UInt16) : long;

3
4 *Description*

5 Converts the value of the specified 16-bit unsigned integer to the equivalent
6 64-bit signed integer.

7 *Return Value:* The 64-bit signed integer equivalent to the value of *value* . A 16-bit
8 unsigned integer.

9 ToInt64

10
11 [C#] public static long ToInt64(uint value);

12 [C++] public: static __int64 ToInt64(unsigned int value);

13 [VB] Public Shared Function ToInt64(ByVal value As UInt32) As Long

14 [JScript] public static function ToInt64(value : UInt32) : long;

15
16 *Description*

17 Converts the value of the specified 32-bit unsigned integer to an equivalent
18 64-bit signed integer.

19 *Return Value:* A 64-bit signed integer equivalent to the value of *value* . A 32-bit
20 unsigned integer.

21 ToInt64

22
23 [C#] public static long ToInt64(ulong value);

24 [C++] public: static __int64 ToInt64(unsigned __int64 value);

25 [VB] Public Shared Function ToInt64(ByVal value As UInt64) As Long

[JScript] public static function ToInt64(value : UInt64) : long;

Description

Converts the value of the specified 64-bit unsigned integer to an equivalent 64-bit signed integer.

Return Value: A 64-bit signed integer equivalent to the value of *value* . A 64-bit unsigned integer.

ToInt64

[C#] public static long ToInt64(object value, IFormatProvider provider);

[C++] public: static __int64 ToInt64(Object* value, IFormatProvider* provider);

[VB] Public Shared Function ToInt64(ByVal value As Object, ByVal provider As IFormatProvider) As Long

[JScript] public static function ToInt64(value : Object, provider : IFormatProvider) : long;

Description

Converts the value of the specified **Object** to a 64-bit signed integer using the specified culture-specific formatting information.

Return Value: A 64-bit signed integer equivalent to the value of *value* , or zero if *value* is **null** .

The return value is the result of invoking the **IConvertible.ToInt64** method of the underlying type of *value* . An **System.Object** that implements the **System.IConvertible** interface. An **System.IFormatProvider** interface implementation that supplies culture-specific formatting information.

ToInt64

```
[C#] public static long ToInt64(string value, IFormatProvider provider);  
[C++] public: static __int64 ToInt64(String* value, IFormatProvider* provider);  
[VB] Public Shared Function ToInt64(ByVal value As String, ByVal provider As  
IFormatProvider) As Long  
[JScript] public static function ToInt64(value : String, provider : IFormatProvider)  
: long;
```

Description

Converts the specified **String** representation of a number to an equivalent 64-bit signed integer using the specified culture-specific formatting information.

Return Value: A 64-bit signed integer equivalent to the value of *value* .

The return value is the result of invoking the **System.Int64.Parse(System.String)** method on *value* . A **System.String** containing a number to convert. An **System.IFormatProvider** interface implementation that supplies culture-specific formatting information.

ToInt64

```
[C#] public static long ToInt64(string value, int fromBase);  
[C++] public: static __int64 ToInt64(String* value, int fromBase);  
[VB] Public Shared Function ToInt64(ByVal value As String, ByVal fromBase As  
Integer) As Long  
[JScript] public static function ToInt64(value : String, fromBase : int) : long;
```

1
2 *Description*

3 Converts the **String** representation of a number in a specified base to an
4 equivalent 64-bit signed integer.

5 *Return Value:* A 64-bit signed integer equivalent to the number in *value* . A
6 **System.String** containing a number. The base of the number in *value*, which must
7 be 2, 8, 10, or 16.

8 ToSByte

9
10 [C#] public static sbyte ToSByte(bool value);

11 [C++] public: static char ToSByte(bool value);

12 [VB] Public Shared Function ToSByte(ByVal value As Boolean) As SByte

13 [JScript] public static function ToSByte(value : Boolean) : SByte;

14
15 *Description*

16 Converts the value of the specified Boolean value to the equivalent 8-bit
17 signed integer.

18 *Return Value:* The number 1 if *value* is **true** ; otherwise, 0. A Boolean value.

19 ToSByte

20
21 [C#] public static sbyte ToSByte(byte value);

22 [C++] public: static char ToSByte(unsigned char value);

23 [VB] Public Shared Function ToSByte(ByVal value As Byte) As SByte

24 [JScript] public static function ToSByte(value : Byte) : SByte;

Description

Converts the value of the specified 8-bit unsigned integer to the equivalent 8-bit signed integer.

Return Value: The 8-bit signed integer equivalent to the value of *value* . An 8-bit unsigned integer.

ToSByte

[C#] public static sbyte ToSByte(char value);

[C++] public: static char ToSByte(__wchar_t value);

[VB] Public Shared Function ToSByte(ByVal value As Char) As SByte

[JScript] public static function ToSByte(value : Char) : SByte;

Description

Converts the value of the specified Unicode character to the equivalent 8-bit signed integer.

Return Value: The 8-bit signed integer equivalent to *value* . A Unicode character.

ToSByte

[C#] public static sbyte ToSByte(DateTime value);

[C++] public: static char ToSByte(DateTime value);

[VB] Public Shared Function ToSByte(ByVal value As DateTime) As SByte

[JScript] public static function ToSByte(value : DateTime) : SByte;

Description

Calling this method always throws **System.InvalidCastException** .

This method is reserved for future use. A **System.DateTime**.

ToSByte

[C#] public static sbyte ToSByte(decimal value);

[C++] public: static char ToSByte(Decimal value);

[VB] Public Shared Function ToSByte(ByVal value As Decimal) As SByte

[JScript] public static function ToSByte(value : Decimal) : SByte;

Description

Converts the value of the specified **Decimal** number to an equivalent 8-bit signed integer.

Return Value: *value* rounded to the nearest 8-bit signed integer. If *value* is halfway between two whole numbers, the even number is returned; that is, 4.5 is converted to 4, and 5.5 is converted to 6. A **System.Decimal** number.

ToSByte

[C#] public static sbyte ToSByte(double value);

[C++] public: static char ToSByte(double value);

[VB] Public Shared Function ToSByte(ByVal value As Double) As SByte

[JScript] public static function ToSByte(value : double) : SByte;

Description

Converts the value of the specified double-precision floating point number to an equivalent 8-bit signed integer.

Return Value: *value* rounded to the nearest 8-bit signed integer. If *value* is halfway between two whole numbers, the even number is returned; that is, 4.5 is converted to 4, and 5.5 is converted to 6. A double-precision floating point number.

ToSByte

[C#] public static sbyte ToSByte(short value);

[C++] public: static char ToSByte(short value);

[VB] Public Shared Function ToSByte(ByVal value As Short) As SByte

[JScript] public static function ToSByte(value : Int16) : SByte;

Description

Converts the value of the specified 16-bit signed integer to the equivalent 8-bit signed integer.

Return Value: The 8-bit signed integer equivalent to the value of *value* . A 16-bit signed integer.

ToSByte

[C#] public static sbyte ToSByte(int value);

[C++] public: static char ToSByte(int value);

[VB] Public Shared Function ToSByte(ByVal value As Integer) As SByte

[JScript] public static function ToSByte(value : int) : SByte;

Description

Converts the value of the specified 32-bit signed integer to an equivalent 8-bit signed integer.

1 *Return Value:* An 8-bit signed integer equivalent to the value of *value* .

3 *Description*

4 Converts the value of the specified 32-bit signed integer to an equivalent 8-
5 bit signed integer.

6 *Return Value:* The 8-bit signed integer equivalent of *value* . A 32-signed integer.

7 ToSByte

9 [C#] public static sbyte ToSByte(long value);

10 [C++] public: static char ToSByte(__int64 value);

11 [VB] Public Shared Function ToSByte(ByVal value As Long) As SByte

12 [JScript] public static function ToSByte(value : long) : SByte;

14 *Description*

15 Converts the value of the specified 64-bit signed integer to an equivalent 8-
16 bit signed integer.

17 *Return Value:* An 8-bit signed integer equivalent to the value of *value* . A 64-bit
18 signed integer.

19 ToSByte

21 [C#] public static sbyte ToSByte(object value);

22 [C++] public: static char ToSByte(Object* value);

23 [VB] Public Shared Function ToSByte(ByVal value As Object) As SByte

24 [JScript] public static function ToSByte(value : Object) : SByte; Converts a
25 specified value to an 8-bit signed integer.

1
2 *Description*

3 Converts the value of the specified **Object** to an 8-bit signed integer.

4 *Return Value:* An 8-bit signed integer equivalent to the value of *value* , or zero if
5 *value* is **null** .

6 The return value is the result of invoking the **IConvertible.ToSByte**
7 method of the underlying type of *value* . An **System.Object** that implements the
8 **System.IConvertible** interface or **null**.

9 ToSByte

10
11 [C#] public static sbyte ToSByte(sbyte value);

12 [C++] public: static char ToSByte(char value);

13 [VB] Public Shared Function ToSByte(ByVal value As SByte) As SByte

14 [JScript] public static function ToSByte(value : SByte) : SByte;

15
16 *Description*

17 Returns the specified 8-bit signed integer; no actual conversion is
18 performed.

19 *Return Value:* *value* is returned unchanged. An 8-bit signed integer.

20 ToSByte

21
22 [C#] public static sbyte ToSByte(float value);

23 [C++] public: static char ToSByte(float value);

24 [VB] Public Shared Function ToSByte(ByVal value As Single) As SByte

25 [JScript] public static function ToSByte(value : float) : SByte;

Description

Converts the value of the specified single-precision floating point number to an equivalent 8-bit signed integer.

Return Value: *value* rounded to the nearest 8-bit signed integer. If *value* is halfway between two whole numbers, the even number is returned; that is, 4.5 is converted to 4, and 5.5 is converted to 6. A single-precision floating point number.

ToSByte

[C#] public static sbyte ToSByte(string value);

[C++] public: static char ToSByte(String* value);

[VB] Public Shared Function ToSByte(ByVal value As String) As SByte

[JScript] public static function ToSByte(value : String) : SByte;

Description

Converts the specified **String** representation of a number to an equivalent 8-bit signed integer.

Return Value: An 8-bit signed integer equivalent to the value of *value* . A

System.String containing a number to convert.

ToSByte

[C#] public static sbyte ToSByte(ushort value);

[C++] public: static char ToSByte(unsigned short value);

[VB] Public Shared Function ToSByte(ByVal value As UInt16) As SByte

[JScript] public static function ToSByte(value : UInt16) : SByte;

1
2 *Description*

3 Converts the value of the specified 16-bit unsigned integer to the equivalent
4 8-bit signed integer.

5 *Return Value:* The 8-bit signed integer equivalent to the value of *value* . A 16-bit
6 unsigned integer.

7 ToSByte

8
9 [C#] public static sbyte ToSByte(uint value);

10 [C++] public: static char ToSByte(unsigned int value);

11 [VB] Public Shared Function ToSByte(ByVal value As UInt32) As SByte

12 [JScript] public static function ToSByte(value : UInt32) : SByte;

13
14 *Description*

15 Converts the value of the specified 32-bit unsigned integer to an equivalent
16 8-bit signed integer.

17 *Return Value:* An 8-bit signed integer equivalent to the value of *value* . A 32-bit
18 unsigned integer.

19 ToSByte

20
21 [C#] public static sbyte ToSByte(ulong value);

22 [C++] public: static char ToSByte(unsigned __int64 value);

23 [VB] Public Shared Function ToSByte(ByVal value As UInt64) As SByte

24 [JScript] public static function ToSByte(value : UInt64) : SByte;

1
2 *Description*

3 Converts the value of the specified 64-bit unsigned integer to an equivalent
4 8-bit signed integer.

5 *Return Value:* An 8-bit signed integer equivalent to the value of *value* . A 64-bit
6 unsigned integer.

7 ToSByte

8
9 [C#] public static sbyte ToSByte(object value, IFormatProvider provider);

10 [C++] public: static char ToSByte(Object* value, IFormatProvider* provider);

11 [VB] Public Shared Function ToSByte(ByVal value As Object, ByVal provider
12 As IFormatProvider) As SByte

13 [JScript] public static function ToSByte(value : Object, provider :
14 IFormatProvider) : SByte;

15
16 *Description*

17 Converts the value of the specified **Object** to an 8-bit signed integer using
18 the specified culture-specific formatting information.

19 *Return Value:* An 8-bit signed integer equivalent to the value of *value* , or zero if
20 *value* is **null** .

21 *provider* enables the user to specify culture-specific conversion information
22 about the contents of *value* . For example, if *value* is a **String** that represents a
23 number, *provider* could supply culture-specific information about the notation
24 used to represent that number. An **System.Object** that implements the
25

System.IConvertible interface. An **System.IFormatProvider** interface implementation that supplies culture-specific formatting information.

ToSByte

[C#] public static sbyte ToSByte(string value, IFormatProvider provider);

[C++] public: static char ToSByte(String* value, IFormatProvider* provider);

[VB] Public Shared Function ToSByte(ByVal value As String, ByVal provider As IFormatProvider) As SByte

[JScript] public static function ToSByte(value : String, provider : IFormatProvider) : SByte;

Description

Converts the specified **String** representation of a number to an equivalent 8-bit signed integer using specified culture-specific formatting information.

Return Value: An 8-bit signed integer equivalent to the value of *value* .

provider is an **IFormatProvider** instance that obtains a **System.Globalization.NumberFormatInfo** object. The **NumberFormatInfo** object provides culture-specific information about the format of *value* . If *provider* is **null** , the **NumberFormatInfo** for the current culture is used. A **System.String** containing a number to convert. An **System.IFormatProvider** interface implementation that supplies culture-specific formatting information.

ToSByte

[C#] public static sbyte ToSByte(string value, int fromBase);

[C++] public: static char ToSByte(String* value, int fromBase);

1 [VB] Public Shared Function ToSByte(ByVal value As String, ByVal fromBase
2 As Integer) As SByte

3 [JScript] public static function ToSByte(value : String, fromBase : int) : SByte;

4
5 *Description*

6 Converts the **String** representation of a number in a specified base to an
7 equivalent 8-bit signed integer.

8 *Return Value:* An 8-bit signed integer equivalent to the number in *value* . A

9 **System.String** containing a number. The base of the number in *value*, which must
10 be 2, 8, 10, or 16.

11 ToSingle

12
13 [C#] public static float ToSingle(bool value);

14 [C++] public: static float ToSingle(bool value);

15 [VB] Public Shared Function ToSingle(ByVal value As Boolean) As Single

16 [JScript] public static function ToSingle(value : Boolean) : float;

17
18 *Description*

19 Converts the value of the specified Boolean value to the equivalent single-
20 precision floating point number.

21 *Return Value:* The number 1 if *value* is **true** ; otherwise, 0. A Boolean value.

22 ToSingle

23
24 [C#] public static float ToSingle(byte value);

25 [C++] public: static float ToSingle(unsigned char value);

1 [VB] Public Shared Function ToSingle(ByVal value As Byte) As Single

2 [JScript] public static function ToSingle(value : Byte) : float;

3
4 *Description*

5 Converts the value of the specified 8-bit unsigned integer to the equivalent
6 single-precision floating point number.

7 *Return Value:* The single-precision floating point number equivalent to the value
8 of *value* . An 8-bit unsigned integer.

9 ToSingle

10
11 [C#] public static float ToSingle(char value);

12 [C++] public: static float ToSingle(__wchar_t value);

13 [VB] Public Shared Function ToSingle(ByVal value As Char) As Single

14 [JScript] public static function ToSingle(value : Char) : float;

15
16 *Description*

17 Calling this method always throws **System.InvalidCastException** .

18 This method is reserved for future use. A Unicode character.

19 ToSingle

20
21 [C#] public static float ToSingle(DateTime value);

22 [C++] public: static float ToSingle(DateTime value);

23 [VB] Public Shared Function ToSingle(ByVal value As DateTime) As Single

24 [JScript] public static function ToSingle(value : DateTime) : float;

1
2 *Description*

3 Calling this method always throws **System.InvalidCastException** .

4 This method is reserved for future use. A **System.DateTime**.

5 ToSingle

6
7 [C#] public static float ToSingle(decimal value);

8 [C++] public: static float ToSingle(Decimal value);

9 [VB] Public Shared Function ToSingle(ByVal value As Decimal) As Single

10 [JScript] public static function ToSingle(value : Decimal) : float;

11
12 *Description*

13 Converts the value of the specified **Decimal** number to an equivalent
14 single-precision floating point number.

15 *Return Value:* A single-precision floating point number equivalent to the value of
16 *value* . A **System.Decimal** number.

17 ToSingle

18
19 [C#] public static float ToSingle(double value);

20 [C++] public: static float ToSingle(double value);

21 [VB] Public Shared Function ToSingle(ByVal value As Double) As Single

22 [JScript] public static function ToSingle(value : double) : float;

23
24 *Description*

Converts the value of the specified double-precision floating point number to an equivalent single-precision floating point number.

Return Value: A single-precision floating point number equivalent to the value of *value* . A double-precision floating point number.

ToSingle

[C#] public static float ToSingle(short value);

[C++] public: static float ToSingle(short value);

[VB] Public Shared Function ToSingle(ByVal value As Short) As Single

[JScript] public static function ToSingle(value : Int16) : float;

Description

Converts the value of the specified 16-bit signed integer to an equivalent single-precision floating point number.

Return Value: A single-precision floating point number equivalent to the value of *value* . A 16-bit signed integer.

ToSingle

[C#] public static float ToSingle(int value);

[C++] public: static float ToSingle(int value);

[VB] Public Shared Function ToSingle(ByVal value As Integer) As Single

[JScript] public static function ToSingle(value : int) : float;

Description

Converts the value of the specified 32-bit signed integer to an equivalent single-precision floating point number.

Return Value: A single-precision floating point number equivalent to the value of *value* . A 32-bit signed integer.

ToSingle

[C#] public static float ToSingle(long value);

[C++] public: static float ToSingle(__int64 value);

[VB] Public Shared Function ToSingle(ByVal value As Long) As Single

[JScript] public static function ToSingle(value : long) : float;

Description

Converts the value of the specified 64-bit signed integer to an equivalent single-precision floating point number.

Return Value: A single-precision floating point number equivalent to the value of *value* . A 64-bit signed integer.

ToSingle

[C#] public static float ToSingle(object value);

[C++] public: static float ToSingle(Object* value);

[VB] Public Shared Function ToSingle(ByVal value As Object) As Single

[JScript] public static function ToSingle(value : Object) : float; Converts a specified value to a single-precision floating point number.

Description

Converts the value of the specified **Object** to a single-precision floating point number.

Return Value: A single-precision floating point number equivalent to the value of *value* , or zero if *value* is **null** .

The return value is the result of invoking the **IConvertible.ToSingle** method of the underlying type of *value* . An **System.Object** that implements the **System.IConvertible** interface or **null**.

ToSingle

[C#] public static float ToSingle(sbyte value);

[C++] public: static float ToSingle(char value);

[VB] Public Shared Function ToSingle(ByVal value As SByte) As Single

[JScript] public static function ToSingle(value : SByte) : float;

Description

Converts the value of the specified 8-bit signed integer to the equivalent single-precision floating point number.

Return Value: The 8-bit signed integer equivalent to the value of *value* . An 8-bit signed integer.

ToSingle

[C#] public static float ToSingle(float value);

[C++] public: static float ToSingle(float value);

[VB] Public Shared Function ToSingle(ByVal value As Single) As Single

[JScript] public static function ToSingle(value : float) : float;

1
2 *Description*

3 Returns the specified single-precision floating point number; no actual
4 conversion is performed.

5 *Return Value:* *value* is returned unchanged. A single-precision floating point
6 number.

7 ToSingle

8
9 [C#] public static float ToSingle(string value);

10 [C++] public: static float ToSingle(String* value);

11 [VB] Public Shared Function ToSingle(ByVal value As String) As Single

12 [JScript] public static function ToSingle(value : String) : float;

13
14 *Description*

15 Converts the specified **String** representation of a number to an equivalent
16 single-precision floating point number.

17 *Return Value:* A single-precision floating point number equivalent to the value of
18 *value* .

19 The return value is the result of invoking the
20 **System.Single.Parse(System.String)** method on *value* . A **System.String**
21 containing a number to convert.

22 ToSingle

23
24 [C#] public static float ToSingle(ushort value);

25 [C++] public: static float ToSingle(unsigned short value);

1 [VB] Public Shared Function ToSingle(ByVal value As UInt16) As Single

2 [JScript] public static function ToSingle(value : UInt16) : float;

3
4 *Description*

5 Converts the value of the specified 16-bit unsigned integer to the equivalent
6 single-precision floating point number.

7 *Return Value:* The single-precision floating point number equivalent to the value
8 of *value* . A 16-bit unsigned integer.

9 ToSingle

10
11 [C#] public static float ToSingle(uint value);

12 [C++] public: static float ToSingle(unsigned int value);

13 [VB] Public Shared Function ToSingle(ByVal value As UInt32) As Single

14 [JScript] public static function ToSingle(value : UInt32) : float;

15
16 *Description*

17 Converts the value of the specified 32-bit unsigned integer to an equivalent
18 single-precision floating point number.

19 *Return Value:* A single-precision floating point number equivalent to the value of
20 *value* . A 32-bit unsigned integer.

21 ToSingle

22
23 [C#] public static float ToSingle(ulong value);

24 [C++] public: static float ToSingle(unsigned __int64 value);

25 [VB] Public Shared Function ToSingle(ByVal value As UInt64) As Single

1 [JScript] public static function ToSingle(value : UInt64) : float;

3 *Description*

4 Converts the value of the specified 64-bit unsigned integer to an equivalent
5 single-precision floating point number.

6 *Return Value:* A single-precision floating point number equivalent to the value of
7 *value* . A 64-bit unsigned integer.

8 ToSingle

10 [C#] public static float ToSingle(object value, IFormatProvider provider);

11 [C++] public: static float ToSingle(Object* value, IFormatProvider* provider);

12 [VB] Public Shared Function ToSingle(ByVal value As Object, ByVal provider
13 As IFormatProvider) As Single

14 [JScript] public static function ToSingle(value : Object, provider :
15 IFormatProvider) : float;

17 *Description*

18 Converts the value of the specified **Object** to an single-precision floating
19 point number using the specified culture-specific formatting information.

20 *Return Value:* A single-precision floating point number equivalent to the value of
21 *value* , or zero if *value* is **null** .

22 The return value is the result of invoking the **IConvertible.ToSingle**
23 method of the underlying type of *value* . An **System.Object** that implements the
24 **System.IConvertible** interface. An **System.IFormatProvider** interface
25 implementation that supplies culture-specific formatting information.

ToSingle

```
[C#] public static float ToSingle(string value, IFormatProvider provider);  
[C++] public: static float ToSingle(String* value, IFormatProvider* provider);  
[VB] Public Shared Function ToSingle(ByVal value As String, ByVal provider As  
IFormatProvider) As Single  
[JScript] public static function ToSingle(value : String, provider :  
IFormatProvider) : float;
```

Description

Converts the specified **String** representation of a number to an equivalent single-precision floating point number using the specified culture-specific formatting information.

Return Value: A single-precision floating point number equivalent to the value of *value* .

The return value is the result of invoking the **System.Single.Parse(System.String)** method on *value* . A **System.String** containing a number to convert. An **System.IFormatProvider** interface implementation that supplies culture-specific formatting information.

ToString

```
[C#] public static string ToString(bool value);  
[C++] public: static String* ToString(bool value);  
[VB] Public Shared Function ToString(ByVal value As Boolean) As String  
[JScript] public static function ToString(value : Boolean) : String;
```

Description

Converts the value of the specified Boolean to its equivalent **String** representation.

Return Value: The **System.String** equivalent of the value of *value* .

This implementation is identical to **System.Boolean.ToString** . A Boolean value.

ToString

[C#] public static string ToString(byte value);

[C++] public: static String* ToString(unsigned char value);

[VB] Public Shared Function ToString(ByVal value As Byte) As String

[JScript] public static function ToString(value : Byte) : String;

Description

Converts the value of the specified 8-bit unsigned integer to its equivalent **String** representation.

Return Value: The **System.String** equivalent of the value of *value* .

This implementation is identical to **System.Byte.ToString** . An 8-bit unsigned integer.

ToString

[C#] public static string ToString(char value);

[C++] public: static String* ToString(__wchar_t value);

[VB] Public Shared Function ToString(ByVal value As Char) As String

1 [JScript] public static function ToString(value : Char) : String;

3 *Description*

4 Converts the value of the specified Unicode character to its equivalent
5 **String** representation.

6 *Return Value:* The **System.String** equivalent of the value of *value* .

7 This implementation is identical to **System.Char.ToString** . A Unicode
8 character.

9 ToString

11 [C#] public static string ToString(DateTime value);

12 [C++] public: static String* ToString(DateTime value);

13 [VB] Public Shared Function ToString(ByVal value As DateTime) As String

14 [JScript] public static function ToString(value : DateTime) : String;

16 *Description*

17 Converts the value of the specified **DateTime** to its equivalent **String**
18 representation.

19 *Return Value:* The **System.String** equivalent of the value of *value* .

20 This implementation is identical to **System.DateTime.ToString** . A
21 **DateTime**.

22 ToString

24 [C#] public static string ToString(decimal value);

25 [C++] public: static String* ToString(Decimal value);

1 [VB] Public Shared Function ToString(ByVal value As Decimal) As String

2 [JScript] public static function ToString(value : Decimal) : String;

3
4 *Description*

5 Converts the value of the specified **Decimal** number to its equivalent

6 **String** representation.

7 *Return Value:* The **System.String** equivalent of the value of *value* .

8 This implementation is identical to **System.Decimal.ToString** . A **Decimal**
9 number.

10 ToString

11
12 [C#] public static string ToString(double value);

13 [C++] public: static String* ToString(double value);

14 [VB] Public Shared Function ToString(ByVal value As Double) As String

15 [JScript] public static function ToString(value : double) : String;

16
17 *Description*

18 Converts the value of the specified double-precision floating point number
19 to its equivalent **String** representation.

20 *Return Value:* The **System.String** equivalent of the value of *value* .

21 This implementation is identical to **System.Double.ToString** . A double-
22 precision floating point number.

23 ToString

24
25 [C#] public static string ToString(short value);

1 [C++] public: static String* ToString(short value);

2 [VB] Public Shared Function ToString(ByVal value As Short) As String

3 [JScript] public static function ToString(value : Int16) : String;

4
5 *Description*

6 Converts the value of the specified 16-bit signed integer to its equivalent

7 **String** representation.

8 *Return Value:* The **System.String** equivalent of the value of *value* .

9 This implementation is identical to **System.Int16.ToString** . A 16-bit
10 signed integer.

11 ToString

12
13 [C#] public static string ToString(int value);

14 [C++] public: static String* ToString(int value);

15 [VB] Public Shared Function ToString(ByVal value As Integer) As String

16 [JScript] public static function ToString(value : int) : String;

17
18 *Description*

19 Converts the value of the specified 32-bit signed integer to its equivalent

20 **String** representation.

21 *Return Value:* The **System.String** equivalent of the value of *value* .

22 This implementation is identical to **System.Int32.ToString** . A 32-bit
23 signed integer.

24 ToString

```

1
2 [C#] public static string ToString(long value);
3 [C++] public: static String* ToString(__int64 value);
4 [VB] Public Shared Function ToString(ByVal value As Long) As String
5 [JScript] public static function ToString(value : long) : String;
6

```

Description

Converts the value of the specified 64-bit signed integer to its equivalent

String representation.

Return Value: The **System.String** equivalent of the value of *value* .

This implementation is identical to **System.Int64.ToString** . A 64-bit signed integer.

ToString

```

15 [C#] public static string ToString(object value);
16 [C++] public: static String* ToString(Object* value);
17 [VB] Public Shared Function ToString(ByVal value As Object) As String
18 [JScript] public static function ToString(value : Object) : String; Converts the
19 specified value to its equivalent String representation.
20

```

Description

Converts the value of the specified **Object** to its **String** representation.

Return Value: The **System.String** representation of the value of *value* , or

System.String.Empty if value is **null** .

1 The return value is the result of invoking the **ToString** method of the
2 underlying type of *value* . An **System.Object** or **null**.

3 **ToString**

4
5 [C#] public static string ToString(sbyte value);

6 [C++] public: static String* ToString(char value);

7 [VB] Public Shared Function ToString(ByVal value As SByte) As String

8 [JScript] public static function ToString(value : SByte) : String;

9
10 *Description*

11 Converts the value of the specified 8-bit signed integer to its equivalent
12 **String** representation.

13 *Return Value:* The **System.String** equivalent of the value of *value* .

14 This implementation is identical to **System.SByte.ToString** . An 8-bit
15 signed integer.

16 **ToString**

17
18 [C#] public static string ToString(float value);

19 [C++] public: static String* ToString(float value);

20 [VB] Public Shared Function ToString(ByVal value As Single) As String

21 [JScript] public static function ToString(value : float) : String;

22
23 *Description*

Converts the value of the specified single-precision floating point number to its equivalent **String** representation.

Return Value: The **System.String** equivalent of the value of *value* .

This implementation is identical to **System.Single.ToString** . A single-precision floating point number.

ToString

[C#] public static string ToString(string value);

[C++] public: static String* ToString(String* value);

[VB] Public Shared Function ToString(ByVal value As String) As String

[JScript] public static function ToString(value : String) : String;

Description

Returns the specified instance of **System.String** ; no actual conversion is performed.

Return Value: *value* is returned unchanged. A **System.String**.

ToString

[C#] public static string ToString(ushort value);

[C++] public: static String* ToString(unsigned short value);

[VB] Public Shared Function ToString(ByVal value As UInt16) As String

[JScript] public static function ToString(value : UInt16) : String;

Description

Converts the value of the specified 16-bit unsigned integer to its equivalent **String** representation.

Return Value: The **System.String** equivalent of the value of *value* .

This implementation is identical to **System.UInt16.ToString** . A 16-bit unsigned integer.

ToString

[C#] public static string ToString(uint value);

[C++] public: static String* ToString(unsigned int value);

[VB] Public Shared Function ToString(ByVal value As UInt32) As String

[JScript] public static function ToString(value : UInt32) : String;

Description

Converts the value of the specified 32-bit unsigned integer to its equivalent **String** representation.

Return Value: The **System.String** equivalent of the value of *value* .

This implementation is identical to **System.UInt32.ToString** . A 32-bit unsigned integer.

ToString

[C#] public static string ToString(ulong value);

[C++] public: static String* ToString(unsigned __int64 value);

[VB] Public Shared Function ToString(ByVal value As UInt64) As String

[JScript] public static function ToString(value : UInt64) : String;

1
2 *Description*

3 Converts the value of the specified 64-bit unsigned integer to its equivalent
4 **String** representation.

5 *Return Value:* The **System.String** equivalent of the value of *value* .

6 This implementation is identical to **System.UInt64.ToString** . A 64-bit
7 unsigned integer.

8 ToString

9
10 [C#] public static string ToString(bool value, IFormatProvider provider);

11 [C++] public: static String* ToString(bool value, IFormatProvider* provider);

12 [VB] Public Shared Function ToString(ByVal value As Boolean, ByVal provider
13 As IFormatProvider) As String

14 [JScript] public static function ToString(value : Boolean, provider :
15 IFormatProvider) : String;

16
17 *Description*

18 Converts the value of the specified Boolean to its equivalent **String**
19 representation.

20 *Return Value:* The **System.String** equivalent of the value of *value* .

21 This implementation is identical to **System.Boolean.ToString** . A Boolean
22 value. (Reserved) An instance of an **System.IFormatProvider** interface
23 implementation.

24 ToString

```

1
2 [C#] public static string ToString(byte value, IFormatProvider provider);
3 [C++] public: static String* ToString(unsigned char value, IFormatProvider*
4 provider);
5 [VB] Public Shared Function ToString(ByVal value As Byte, ByVal provider As
6 IFormatProvider) As String
7 [JScript] public static function ToString(value : Byte, provider : IFormatProvider)
8 : String;
9

```

Description

Converts the value of the specified 8-bit unsigned integer to its equivalent

String representation.

Return Value: The **System.String** equivalent of the value of *value* .

This implementation is identical to **System.Byte.ToString** . An 8-bit unsigned integer. An **System.IFormatProvider** interface implementation that supplies culture-specific formatting information.

ToString

```

17
18
19 [C#] public static string ToString(byte value, int toBase);
20 [C++] public: static String* ToString(unsigned char value, int toBase);
21 [VB] Public Shared Function ToString(ByVal value As Byte, ByVal toBase As
22 Integer) As String
23 [JScript] public static function ToString(value : Byte, toBase : int) : String;
24

```

Description

Converts the value of an 8-bit unsigned integer to its equivalent **String** representation in a specified base.

Return Value: The **String** representation of *value* in base *toBase* . An 8-bit unsigned integer. The base of the return value, which must be 2, 8, 10, or 16.

ToString

[C#] public static string ToString(char value, IFormatProvider provider);

[C++] public: static String* ToString(__wchar_t value, IFormatProvider* provider);

[VB] Public Shared Function ToString(ByVal value As Char, ByVal provider As IFormatProvider) As String

[JScript] public static function ToString(value : Char, provider : IFormatProvider) : String;

Description

Converts the value of the specified Unicode character to its equivalent **String** representation.

Return Value: The **System.String** equivalent of the value of *value* .

This implementation is identical to **System.Char.ToString** . A Unicode character. An **System.IFormatProvider** interface implementation that supplies culture-specific formatting information.

ToString

[C#] public static string ToString(DateTime value, IFormatProvider provider);

[C++] public: static String* ToString(DateTime value, IFormatProvider*

provider);

[VB] Public Shared Function ToString(ByVal value As DateTime, ByVal provider
As IFormatProvider) As String

[JScript] public static function ToString(value : DateTime, provider :
IFormatProvider) : String;

Description

Converts the value of the specified **DateTime** to its equivalent **String**
representation.

Return Value: The **System.String** equivalent of the value of *value* .

This implementation is identical to **System.DateTime.ToString** . A
DateTime. An **System.IFormatProvider** interface implementation that supplies
culture-specific formatting information.

ToString

[C#] public static string ToString(decimal value, IFormatProvider provider);

[C++] public: static String* ToString(Decimal value, IFormatProvider* provider);

[VB] Public Shared Function ToString(ByVal value As Decimal, ByVal provider
As IFormatProvider) As String

[JScript] public static function ToString(value : Decimal, provider :
IFormatProvider) : String;

Description

Converts the value of the specified **Decimal** number to its equivalent **String** representation.

Return Value: The **System.String** equivalent of the value of *value* .

This implementation is identical to **System.Decimal.ToString** . A **Decimal** number. An **System.IFormatProvider** interface implementation that supplies culture-specific formatting information.

ToString

[C#] public static string ToString(double value, IFormatProvider provider);

[C++] public: static String* ToString(double value, IFormatProvider* provider);

[VB] Public Shared Function ToString(ByVal value As Double, ByVal provider As IFormatProvider) As String

[JScript] public static function ToString(value : double, provider : IFormatProvider) : String;

Description

Converts the value of the specified double-precision floating point number to its equivalent **String** representation.

Return Value: The **System.String** equivalent of the value of *value* .

This implementation is identical to **System.Double.ToString** . A double-precision floating point number. An **System.IFormatProvider** interface implementation that supplies culture-specific formatting information.

ToString

[C#] public static string ToString(short value, IFormatProvider provider);


```

1 [C++] public: static String* ToString(short value, IFormatProvider* provider);
2 [VB] Public Shared Function ToString(ByVal value As Short, ByVal provider As
3 IFormatProvider) As String
4 [JScript] public static function ToString(value : Int16, provider : IFormatProvider)
5 : String;
6

```

Description

Converts the value of the specified 16-bit signed integer to its equivalent **String** representation.

Return Value: The **System.String** equivalent of the value of *value* .

This implementation is identical to **System.Int16.ToString** . A 16-bit signed integer. An **System.IFormatProvider** interface implementation that supplies culture-specific formatting information.

ToString

```

16 [C#] public static string ToString(short value, int toBase);
17 [C++] public: static String* ToString(short value, int toBase);
18 [VB] Public Shared Function ToString(ByVal value As Short, ByVal toBase As
19 Integer) As String
20 [JScript] public static function ToString(value : Int16, toBase : int) : String;
21

```

Description

Converts the value of a 16-bit signed integer to its equivalent **String** representation in a specified base.

Return Value: The **String** representation of *value* in base *toBase* . A 16-bit signed integer. The base of the return value, which must be 2, 8, 10, or 16.

ToString

[C#] public static string ToString(int value, IFormatProvider provider);

[C++] public: static String* ToString(int value, IFormatProvider* provider);

[VB] Public Shared Function ToString(ByVal value As Integer, ByVal provider As IFormatProvider) As String

[JScript] public static function ToString(value : int, provider : IFormatProvider) : String;

Description

Converts the value of the specified 32-bit signed integer to its equivalent **String** representation.

Return Value: The **System.String** equivalent of the value of *value* .

This implementation is identical to **System.Int32.ToString** . A 32-bit signed integer. An **System.IFormatProvider** interface implementation that supplies culture-specific formatting information.

ToString

[C#] public static string ToString(int value, int toBase);

[C++] public: static String* ToString(int value, int toBase);

[VB] Public Shared Function ToString(ByVal value As Integer, ByVal toBase As Integer) As String

[JScript] public static function ToString(value : int, toBase : int) : String;

1
2 *Description*

3 Converts the value of a 32-bit signed integer to its equivalent **String**
4 representation in a specified base.

5 *Return Value:* The **String** representation of *value* in base *toBase* . A 32-bit signed
6 integer. The base of the return value, which must be 2, 8, 10, or 16.

7 ToString

8
9 [C#] public static string ToString(long value, IFormatProvider provider);

10 [C++] public: static String* ToString(__int64 value, IFormatProvider* provider);

11 [VB] Public Shared Function ToString(ByVal value As Long, ByVal provider As
12 IFormatProvider) As String

13 [JScript] public static function ToString(value : long, provider : IFormatProvider) :
14 String;

15
16 *Description*

17 Converts the value of the specified 64-bit signed integer to its equivalent
18 **String** representation.

19 *Return Value:* The **System.String** equivalent of the value of *value* .

20 This implementation is identical to **System.Int64.ToString** . A 64-bit
21 signed integer. An **System.IFormatProvider** interface implementation that
22 supplies culture-specific formatting information.

23 ToString

24
25 [C#] public static string ToString(long value, int toBase);

1 [C++] public: static String* ToString(__int64 value, int toBase);

2 [VB] Public Shared Function ToString(ByVal value As Long, ByVal toBase As
3 Integer) As String

4 [JScript] public static function ToString(value : long, toBase : int) : String;

5
6 *Description*

7 Converts the value of a 64-bit signed integer to its equivalent **String**
8 representation in a specified base.

9 *Return Value:* The **String** representation of *value* in base *toBase* . A 64-bit signed
10 integer. The base of the return value, which must be 2, 8, 10, or 16.

11 **ToString**

12
13 [C#] public static string ToString(object value, IFormatProvider provider);

14 [C++] public: static String* ToString(Object* value, IFormatProvider* provider);

15 [VB] Public Shared Function ToString(ByVal value As Object, ByVal provider
16 As IFormatProvider) As String

17 [JScript] public static function ToString(value : Object, provider :
18 IFormatProvider) : String;

19
20 *Description*

21 Converts the value of the specified **Object** to its equivalent **String**
22 representation using the specified culture-specific formatting information.

23 *Return Value:* The **System.String** representation of the value of *value* , or
24 **System.String.Empty** if *value* is **null** .

provider enables the user to specify culture-specific conversion information about the contents of *value* . For example, if *value* is a **String** that represents a number, *provider* could supply culture-specific information about the notation used to represent that number. An **System.Object** or **null**. An **System.IFormatProvider** interface implementation that supplies culture-specific formatting information.

ToString

[C#] public static string ToString(sbyte value, IFormatProvider provider);
[C++] public: static String* ToString(char value, IFormatProvider* provider);
[VB] Public Shared Function ToString(ByVal value As SByte, ByVal provider As IFormatProvider) As String
[JScript] public static function ToString(value : SByte, provider : IFormatProvider) : String;

Description

Converts the value of the specified 8-bit signed integer to its equivalent **String** representation.

Return Value: The **System.String** equivalent of the value of *value* .

This implementation is identical to **System.SByte.ToString** . An 8-bit signed integer. An **System.IFormatProvider** interface implementation that supplies culture-specific formatting information.

ToString

[C#] public static string ToString(float value, IFormatProvider provider);

```

1 [C++] public: static String* ToString(float value, IFormatProvider* provider);
2 [VB] Public Shared Function ToString(ByVal value As Single, ByVal provider As
3 IFormatProvider) As String
4 [JScript] public static function ToString(value : float, provider : IFormatProvider)
5 : String;
6

```

Description

Converts the value of the specified single-precision floating point number to its equivalent **String** representation.

Return Value: The **System.String** equivalent of the value of *value* .

This implementation is identical to **System.Single.ToString** . A single-precision floating point number. An **System.IFormatProvider** interface implementation that supplies culture-specific formatting information.

ToString

```

16 [C#] public static string ToString(string value, IFormatProvider provider);
17 [C++] public: static String* ToString(String* value, IFormatProvider* provider);
18 [VB] Public Shared Function ToString(ByVal value As String, ByVal provider As
19 IFormatProvider) As String
20 [JScript] public static function ToString(value : String, provider :
21 IFormatProvider) : String;
22

```

Description

1 Returns the specified instance of **System.String** ; no actual conversion is
2 performed.

3 *Return Value:* *value* is returned unchanged.

4 This method ignores the *provider* parameter. A **System.String**. An
5 **System.IFormatProvider** interface implementation that supplies culture-specific
6 formatting information.

7 ToString

8
9 [C#] public static string ToString(ushort value, IFormatProvider provider);

10 [C++] public: static String* ToString(unsigned short value, IFormatProvider*
11 provider);

12 [VB] Public Shared Function ToString(ByVal value As UInt16, ByVal provider
13 As IFormatProvider) As String

14 [JScript] public static function ToString(value : UInt16, provider :
15 IFormatProvider) : String;

16
17 *Description*

18 Converts the value of the specified 16-bit unsigned integer to its equivalent
19 **String** representation.

20 *Return Value:* The **System.String** equivalent of the value of *value* .

21 This implementation is identical to **System.UInt16.ToString** . A 16-bit
22 unsigned integer. An **System.IFormatProvider** interface implementation that
23 supplies culture-specific formatting information.

24 ToString

```

1
2 [C#] public static string ToString(uint value, IFormatProvider provider);
3 [C++] public: static String* ToString(unsigned int value, IFormatProvider*
4 provider);
5 [VB] Public Shared Function ToString(ByVal value As UInt32, ByVal provider
6 As IFormatProvider) As String
7 [JScript] public static function ToString(value : UInt32, provider :
8 IFormatProvider) : String;
9

```

Description

Converts the value of the specified 32-bit unsigned integer to its equivalent

String representation.

Return Value: The **System.String** equivalent of the value of *value* .

This implementation is identical to **System.UInt32.ToString** . A 32-bit unsigned integer. An **System.IFormatProvider** interface implementation that supplies culture-specific formatting information.

ToString

```

18
19 [C#] public static string ToString(ulong value, IFormatProvider provider);
20 [C++] public: static String* ToString(unsigned __int64 value, IFormatProvider*
21 provider);
22 [VB] Public Shared Function ToString(ByVal value As UInt64, ByVal provider
23 As IFormatProvider) As String
24 [JScript] public static function ToString(value : UInt64, provider :
25 IFormatProvider) : String;

```


1
2 *Description*

3 Converts the value of the specified 64-bit unsigned integer to its equivalent
4 **String** representation.

5 *Return Value:* The **System.String** equivalent of the value of *value* .

6 This implementation is identical to **System.UInt64.ToString** . A 64-bit
7 unsigned integer. An **System.IFormatProvider** interface implementation that
8 supplies culture-specific formatting information.

9 ToUInt16

10
11 [C#] public static ushort ToUInt16(bool value);

12 [C++] public: static unsigned short ToUInt16(bool value);

13 [VB] Public Shared Function ToUInt16(ByVal value As Boolean) As UInt16

14 [JScript] public static function ToUInt16(value : Boolean) : UInt16;

15
16 *Description*

17 Converts the value of the specified Boolean value to the equivalent 16-bit
18 unsigned integer.

19 *Return Value:* The number 1 if *value* is **true** ; otherwise, 0. A Boolean value.

20 ToUInt16

21
22 [C#] public static ushort ToUInt16(byte value);

23 [C++] public: static unsigned short ToUInt16(unsigned char value);

24 [VB] Public Shared Function ToUInt16(ByVal value As Byte) As UInt16

25 [JScript] public static function ToUInt16(value : Byte) : UInt16;

1
2 *Description*

3 Converts the value of the specified 8-bit unsigned integer to the equivalent
4 16-bit unsigned integer.

5 *Return Value:* The 16-bit unsigned integer equivalent to the value of *value* . An 8-
6 bit unsigned integer.

7 ToUInt16

8
9 [C#] public static ushort ToUInt16(char value);

10 [C++] public: static unsigned short ToUInt16(__wchar_t value);

11 [VB] Public Shared Function ToUInt16(ByVal value As Char) As UInt16

12 [JScript] public static function ToUInt16(value : Char) : UInt16;

13
14 *Description*

15 Converts the value of the specified Unicode character to the equivalent 16-
16 bit unsigned integer.

17 *Return Value:* The 16-bit unsigned integer equivalent to *value* . A Unicode
18 character.

19 ToUInt16

20
21 [C#] public static ushort ToUInt16(DateTime value);

22 [C++] public: static unsigned short ToUInt16(DateTime value);

23 [VB] Public Shared Function ToUInt16(ByVal value As DateTime) As UInt16

24 [JScript] public static function ToUInt16(value : DateTime) : UInt16;

1
2 *Description*

3 Calling this method always throws **System.InvalidCastException** .

4 This method is reserved for future use. A **System.DateTime**.

5 ToUInt16

6
7 [C#] public static ushort ToUInt16(decimal value);

8 [C++] public: static unsigned short ToUInt16(Decimal value);

9 [VB] Public Shared Function ToUInt16(ByVal value As Decimal) As UInt16

10 [JScript] public static function ToUInt16(value : Decimal) : UInt16;

11
12 *Description*

13 Converts the value of the specified **Decimal** number to an equivalent 16-bit
14 unsigned integer.

15 *Return Value:* *value* rounded to the nearest 16-bit unsigned integer. If *value* is
16 halfway between two whole numbers, the even number is returned; that is, 4.5 is
17 converted to 4, and 5.5 is converted to 6. A **System.Decimal** number.

18 ToUInt16

19
20 [C#] public static ushort ToUInt16(double value);

21 [C++] public: static unsigned short ToUInt16(double value);

22 [VB] Public Shared Function ToUInt16(ByVal value As Double) As UInt16

23 [JScript] public static function ToUInt16(value : double) : UInt16;

24
25 *Description*

Converts the value of the specified double-precision floating point number to an equivalent 16-bit unsigned integer.

Return Value: *value* rounded to the nearest 16-bit unsigned integer. If *value* is halfway between two whole numbers, the even number is returned; that is, 4.5 is converted to 4, and 5.5 is converted to 6. A double-precision floating point number.

ToUInt16

[C#] public static ushort ToUInt16(short value);

[C++] public: static unsigned short ToUInt16(short value);

[VB] Public Shared Function ToUInt16(ByVal value As Short) As UInt16

[JScript] public static function ToUInt16(value : Int16) : UInt16;

Description

Converts the value of the specified 16-bit signed integer to the equivalent 16-bit unsigned integer.

Return Value: The 16-bit unsigned integer equivalent to the value of *value* . A 16-bit signed integer.

ToUInt16

[C#] public static ushort ToUInt16(int value);

[C++] public: static unsigned short ToUInt16(int value);

[VB] Public Shared Function ToUInt16(ByVal value As Integer) As UInt16

[JScript] public static function ToUInt16(value : int) : UInt16;

1
2 *Description*

3 Converts the value of the specified 32-bit signed integer to an equivalent
4 16-bit unsigned integer.

5 *Return Value:* The 16-bit unsigned integer equivalent of *value* . A 32-bit signed
6 integer.

7 ToUInt16

8
9 [C#] public static ushort ToUInt16(long value);

10 [C++] public: static unsigned short ToUInt16(__int64 value);

11 [VB] Public Shared Function ToUInt16(ByVal value As Long) As UInt16

12 [JScript] public static function ToUInt16(value : long) : UInt16;

13
14 *Description*

15 Converts the value of the specified 64-bit signed integer to an equivalent
16 16-bit unsigned integer.

17 *Return Value:* A 16-bit unsigned integer equivalent to the value of *value* . A 64-bit
18 signed integer.

19 ToUInt16

20
21 [C#] public static ushort ToUInt16(object value);

22 [C++] public: static unsigned short ToUInt16(Object* value);

23 [VB] Public Shared Function ToUInt16(ByVal value As Object) As UInt16

24 [JScript] public static function ToUInt16(value : Object) : UInt16; Converts a
25 specified value to a 16-bit unsigned integer.

1
2 *Description*

3 Converts the value of the specified **Object** to a 16-bit unsigned integer.

4 *Return Value:* A 16-bit unsigned integer equivalent to the value of *value* , or zero
5 if *value* is **null** .

6 The return value is the result of invoking the **IConvertible.ToInt16**
7 method of the underlying type of *value* . An **System.Object** that implements the
8 **System.IConvertible** interface or **null**.

9 ToUInt16

10
11 [C#] public static ushort ToUInt16(sbyte value);

12 [C++] public: static unsigned short ToUInt16(char value);

13 [VB] Public Shared Function ToUInt16(ByVal value As SByte) As UInt16

14 [JScript] public static function ToUInt16(value : SByte) : UInt16;

15
16 *Description*

17 Converts the value of the specified 8-bit signed integer to the equivalent 16-
18 bit unsigned integer.

19 *Return Value:* The 8-bit unsigned integer equivalent to the value of *value* . An 8-
20 bit signed integer.

21 ToUInt16

22
23 [C#] public static ushort ToUInt16(float value);

24 [C++] public: static unsigned short ToUInt16(float value);

25 [VB] Public Shared Function ToUInt16(ByVal value As Single) As UInt16

1 [JScript] public static function ToUInt16(value : float) : UInt16;

3 *Description*

4 Converts the value of the specified single-precision floating point number
5 to an equivalent 16-bit unsigned integer.

6 *Return Value:* *value* rounded to the nearest 16-bit unsigned integer. If *value* is
7 halfway between two whole numbers, the even number is returned; that is, 4.5 is
8 converted to 4, and 5.5 is converted to 6. A single-precision floating point number.

9 ToUInt16

11 [C#] public static ushort ToUInt16(string value);

12 [C++] public: static unsigned short ToUInt16(String* value);

13 [VB] Public Shared Function ToUInt16(ByVal value As String) As UInt16

14 [JScript] public static function ToUInt16(value : String) : UInt16;

16 *Description*

17 Converts the specified **String** representation of a number to an equivalent
18 16-bit unsigned integer.

19 *Return Value:* A 16-bit unsigned integer equivalent to the value of *value* .

20 The return value is the result of invoking
21 **System.UInt16.Parse(System.String)** on *value* . A **System.String** containing a
22 number to convert.

23 ToUInt16

25 [C#] public static ushort ToUInt16(ushort value);

```

1 [C++] public: static unsigned short ToUInt16(unsigned short value);
2 [VB] Public Shared Function ToUInt16(ByVal value As UInt16) As UInt16
3 [JScript] public static function ToUInt16(value : UInt16) : UInt16;
4

```

Description

Returns the specified 16-bit unsigned integer; no actual conversion is performed.

Return Value: *value* is returned unchanged. A 16-bit unsigned integer.

ToUInt16

```

11 [C#] public static ushort ToUInt16(uint value);
12 [C++] public: static unsigned short ToUInt16(unsigned int value);
13 [VB] Public Shared Function ToUInt16(ByVal value As UInt32) As UInt16
14 [JScript] public static function ToUInt16(value : UInt32) : UInt16;
15

```

Description

Converts the value of the specified 32-bit unsigned integer to an equivalent 16-bit unsigned integer.

Return Value: A 16-bit unsigned integer equivalent to the value of *value* . A 32-bit unsigned integer.

ToUInt16

```

23 [C#] public static ushort ToUInt16(ulong value);
24 [C++] public: static unsigned short ToUInt16(unsigned __int64 value);
25 [VB] Public Shared Function ToUInt16(ByVal value As UInt64) As UInt16

```


1 [JScript] public static function ToUInt16(value : UInt64) : UInt16;

3 *Description*

4 Converts the value of the specified 64-bit unsigned integer to an equivalent
5 16-bit unsigned integer.

6 *Return Value:* A 16-bit unsigned integer equivalent to the value of *value* . A 64-bit
7 unsigned integer.

8 ToUInt16

9
10 [C#] public static ushort ToUInt16(object value, IFormatProvider provider);

11 [C++] public: static unsigned short ToUInt16(Object* value, IFormatProvider*
12 provider);

13 [VB] Public Shared Function ToUInt16(ByVal value As Object, ByVal provider
14 As IFormatProvider) As UInt16

15 [JScript] public static function ToUInt16(value : Object, provider :
16 IFormatProvider) : UInt16;

17
18 *Description*

19 Converts the value of the specified **Object** to a 16-bit unsigned integer
20 using the specified culture-specific formatting information.

21 *Return Value:* A 16-bit unsigned integer equivalent to the value of *value* , or zero
22 if *value* is **null** .

23 The return value is the result of invoking the **IConvertible.ToUInt16**
24 method of the underlying type of *value* . An **System.Object** that implements the

1 **System.IConvertible** interface. An **System.IFormatProvider** interface
2 implementation that supplies culture-specific formatting information.

3 ToUInt16

4
5 [C#] public static ushort ToUInt16(string value, IFormatProvider provider);

6 [C++] public: static unsigned short ToUInt16(String* value, IFormatProvider*
7 provider);

8 [VB] Public Shared Function ToUInt16(ByVal value As String, ByVal provider
9 As IFormatProvider) As UInt16

10 [JScript] public static function ToUInt16(value : String, provider :
11 IFormatProvider) : UInt16;

12
13 *Description*

14 Converts the specified **String** representation of a number to an equivalent
15 16-bit unsigned integer using specified culture-specific formatting information.

16 *Return Value:* A 16-bit unsigned integer equivalent to the value of *value* .

17 The return value is the result of invoking
18 **System.UInt16.Parse(System.String)** on *value* . A **System.String** containing a
19 number to convert. An **System.IFormatProvider** interface implementation that
20 supplies culture-specific formatting information.

21 ToUInt16

22
23 [C#] public static ushort ToUInt16(string value, int fromBase);

24 [C++] public: static unsigned short ToUInt16(String* value, int fromBase);

25 [VB] Public Shared Function ToUInt16(ByVal value As String, ByVal fromBase

As Integer) As UInt16

[JScript] public static function ToUInt16(value : String, fromBase : int) : UInt16;

Description

Converts the **String** representation of a number in a specified base to an equivalent 16-bit unsigned integer.

Return Value: A 16-bit unsigned integer equivalent to the number in *value* . A

System.String containing a number. The base of the number in *value*, which must be 2, 8, 10, or 16.

ToUInt32

[C#] public static uint ToUInt32(bool value);

[C++] public: static unsigned int ToUInt32(bool value);

[VB] Public Shared Function ToUInt32(ByVal value As Boolean) As UInt32

[JScript] public static function ToUInt32(value : Boolean) : UInt32;

Description

Converts the value of the specified Boolean value to the equivalent 32-bit unsigned integer.

Return Value: The number 1 if *value* is **true** ; otherwise, 0. A Boolean value.

ToUInt32

[C#] public static uint ToUInt32(byte value);

[C++] public: static unsigned int ToUInt32(unsigned char value);

[VB] Public Shared Function ToUInt32(ByVal value As Byte) As UInt32

1 [JScript] public static function ToUInt32(value : Byte) : UInt32;

3 *Description*

4 Converts the value of the specified 8-bit unsigned integer to the equivalent
5 32-bit signed integer.

6 *Return Value:* The 32-bit signed integer equivalent to the value of *value* . An 8-bit
7 unsigned integer.

8 ToUInt32

10 [C#] public static uint ToUInt32(char value);

11 [C++] public: static unsigned int ToUInt32(__wchar_t value);

12 [VB] Public Shared Function ToUInt32(ByVal value As Char) As UInt32

13 [JScript] public static function ToUInt32(value : Char) : UInt32;

15 *Description*

16 Converts the value of the specified Unicode character to the equivalent 32-
17 bit unsigned integer.

18 *Return Value:* The 32-bit unsigned integer equivalent to *value* . A Unicode
19 character.

20 ToUInt32

22 [C#] public static uint ToUInt32(DateTime value);

23 [C++] public: static unsigned int ToUInt32(DateTime value);

24 [VB] Public Shared Function ToUInt32(ByVal value As DateTime) As UInt32

25 [JScript] public static function ToUInt32(value : DateTime) : UInt32;

1
2 *Description*

3 Calling this method always throws **System.InvalidCastException** .

4 This method is reserved for future use. A **System.DateTime**.

5 ToUInt32

6
7 [C#] public static uint ToUInt32(decimal value);

8 [C++] public: static unsigned int ToUInt32(Decimal value);

9 [VB] Public Shared Function ToUInt32(ByVal value As Decimal) As UInt32

10 [JScript] public static function ToUInt32(value : Decimal) : UInt32;

11
12 *Description*

13 Converts the value of the specified **Decimal** number to an equivalent 32-bit
14 unsigned integer.

15 *Return Value:* *value* rounded to the nearest 32-bit unsigned integer. If *value* is
16 halfway between two whole numbers, the even number is returned; that is, 4.5 is
17 converted to 4, and 5.5 is converted to 6. A **System.Decimal** number.

18 ToUInt32

19
20 [C#] public static uint ToUInt32(double value);

21 [C++] public: static unsigned int ToUInt32(double value);

22 [VB] Public Shared Function ToUInt32(ByVal value As Double) As UInt32

23 [JScript] public static function ToUInt32(value : double) : UInt32;

24
25 *Description*

Converts the value of the specified double-precision floating point number to an equivalent 32-bit unsigned integer.

Return Value: *value* rounded to the nearest 32-bit unsigned integer. If *value* is halfway between two whole numbers, the even number is returned; that is, 4.5 is converted to 4, and 5.5 is converted to 6. A double-precision floating point number.

ToUInt32

[C#] public static uint ToUInt32(short value);

[C++] public: static unsigned int ToUInt32(short value);

[VB] Public Shared Function ToUInt32(ByVal value As Short) As UInt32

[JScript] public static function ToUInt32(value : Int16) : UInt32;

Description

Converts the value of the specified 16-bit signed integer to the equivalent 32-bit unsigned integer.

Return Value: The 32-bit unsigned integer equivalent to the value of *value* . A 32-bit signed integer.

ToUInt32

[C#] public static uint ToUInt32(int value);

[C++] public: static unsigned int ToUInt32(int value);

[VB] Public Shared Function ToUInt32(ByVal value As Integer) As UInt32

[JScript] public static function ToUInt32(value : int) : UInt32;

1
2 *Description*

3 Converts the value of the specified 32-bit signed integer to an equivalent
4 32-bit unsigned integer.

5 *Return Value:* The 32-bit unsigned integer equivalent of *value* . A 32-bit signed
6 integer.

7 ToUInt32

8
9 [C#] public static uint ToUInt32(long value);

10 [C++] public: static unsigned int ToUInt32(__int64 value);

11 [VB] Public Shared Function ToUInt32(ByVal value As Long) As UInt32

12 [JScript] public static function ToUInt32(value : long) : UInt32;

13
14 *Description*

15 Converts the value of the specified 64-bit signed integer to an equivalent
16 32-bit unsigned integer.

17 *Return Value:* A 32-bit unsigned integer equivalent to the value of *value* . A 64-bit
18 signed integer.

19 ToUInt32

20
21 [C#] public static uint ToUInt32(object value);

22 [C++] public: static unsigned int ToUInt32(Object* value);

23 [VB] Public Shared Function ToUInt32(ByVal value As Object) As UInt32

24 [JScript] public static function ToUInt32(value : Object) : UInt32; Converts a
25 specified value to a 32-bit unsigned integer.

Description

Converts the value of the specified **Object** to a 32-bit unsigned integer.

Return Value: A 32-bit unsigned integer equivalent to the value of *value* , or zero if *value* is **null** .

The return value is the result of invoking the **ICollection.ToUInt32** method of the underlying type of *value* . An **System.Object** that implements the **System.IConvertible** interface or **null**.

ToUInt32

[C#] public static uint ToUInt32(sbyte value);

[C++] public: static unsigned int ToUInt32(char value);

[VB] Public Shared Function ToUInt32(ByVal value As SByte) As UInt32

[JScript] public static function ToUInt32(value : SByte) : UInt32;

Description

Converts the value of the specified 8-bit signed integer to the equivalent 32-bit unsigned integer.

Return Value: The 8-bit unsigned integer equivalent to the value of *value* . An 8-bit signed integer.

ToUInt32

[C#] public static uint ToUInt32(float value);

[C++] public: static unsigned int ToUInt32(float value);

[VB] Public Shared Function ToUInt32(ByVal value As Single) As UInt32

1 [JScript] public static function ToUInt32(value : float) : UInt32;

3 *Description*

4 Converts the value of the specified single-precision floating point number
5 to an equivalent 32-bit unsigned integer.

6 *Return Value:* *value* rounded to the nearest 32-bit unsigned integer. If *value* is
7 halfway between two whole numbers, the even number is returned; that is, 4.5 is
8 converted to 4, and 5.5 is converted to 6. A single-precision floating point number.

9 ToUInt32

11 [C#] public static uint ToUInt32(string value);

12 [C++] public: static unsigned int ToUInt32(String* value);

13 [VB] Public Shared Function ToUInt32(ByVal value As String) As UInt32

14 [JScript] public static function ToUInt32(value : String) : UInt32;

16 *Description*

17 Converts the specified **String** representation of a number to an equivalent
18 32-bit signed integer.

19 *Return Value:* A 32-bit signed integer equivalent to the value of *value* .

20 The return value is the result of invoking the
21 **System.Int32.Parse(System.String)** method on *value* . A **System.String**
22 containing a number to convert.

23 ToUInt32

25 [C#] public static uint ToUInt32(ushort value);

1 [C++] public: static unsigned int ToUInt32(unsigned short value);

2 [VB] Public Shared Function ToUInt32(ByVal value As UInt16) As UInt32

3 [JScript] public static function ToUInt32(value : UInt16) : UInt32;

4
5 *Description*

6 Converts the value of the specified 16-bit unsigned integer to the equivalent
7 32-bit unsigned integer.

8 *Return Value:* The 32-bit unsigned integer equivalent to the value of *value* . A 32-
9 bit signed integer.

10 ToUInt32

11
12 [C#] public static uint ToUInt32(uint value);

13 [C++] public: static unsigned int ToUInt32(unsigned int value);

14 [VB] Public Shared Function ToUInt32(ByVal value As UInt32) As UInt32

15 [JScript] public static function ToUInt32(value : UInt32) : UInt32;

16
17 *Description*

18 Returns the specified 32-bit unsigned integer; no actual conversion is
19 performed.

20 *Return Value:* *value* is returned unchanged. A 32-bit unsigned integer.

21 ToUInt32

22
23 [C#] public static uint ToUInt32(ulong value);

24 [C++] public: static unsigned int ToUInt32(unsigned __int64 value);

25 [VB] Public Shared Function ToUInt32(ByVal value As UInt64) As UInt32

1 [JScript] public static function ToUInt32(value : UInt64) : UInt32;

3 *Description*

4 Converts the value of the specified 64-bit unsigned integer to an equivalent
5 32-bit unsigned integer.

6 *Return Value:* A 32-bit unsigned integer equivalent to the value of *value* . A 64-bit
7 unsigned integer.

8 ToUInt32

10 [C#] public static uint ToUInt32(object value, IFormatProvider provider);

11 [C++] public: static unsigned int ToUInt32(Object* value, IFormatProvider*
12 provider);

13 [VB] Public Shared Function ToUInt32(ByVal value As Object, ByVal provider
14 As IFormatProvider) As UInt32

15 [JScript] public static function ToUInt32(value : Object, provider :
16 IFormatProvider) : UInt32;

18 *Description*

19 Converts the value of the specified **Object** to a 32-bit unsigned integer
20 using the specified culture-specific formatting information.

21 *Return Value:* A 32-bit unsigned integer equivalent to the value of *value* , or zero
22 if *value* is **null** .

23 The return value is the result of invoking the **IConvertible.ToUInt32**
24 method of the underlying type of *value* . An **System.Object** that implements the
25

System.IConvertible interface. An **System.IFormatProvider** interface implementation that supplies culture-specific formatting information.

ToUInt32

[C#] public static uint ToUInt32(string value, IFormatProvider provider);

[C++] public: static unsigned int ToUInt32(String* value, IFormatProvider* provider);

[VB] Public Shared Function ToUInt32(ByVal value As String, ByVal provider As IFormatProvider) As UInt32

[JScript] public static function ToUInt32(value : String, provider : IFormatProvider) : UInt32;

Description

Converts the specified **String** representation of a number to an equivalent 32-bit unsigned integer using the specified culture-specific formatting information.

Return Value: A 32-bit unsigned integer equivalent to the value of *value* .

The return value is the result of invoking **System.UInt32.Parse(System.String)** on *value* . A **System.String** containing a number to convert. An **System.IFormatProvider** interface implementation that supplies culture-specific formatting information.

ToUInt32

[C#] public static uint ToUInt32(string value, int fromBase);

[C++] public: static unsigned int ToUInt32(String* value, int fromBase);

[VB] Public Shared Function ToUInt32(ByVal value As String, ByVal fromBase

As Integer) As UInt32

[JScript] public static function ToUInt32(value : String, fromBase : int) : UInt32;

Description

Converts the **String** representation of a number in a specified base to an equivalent 32-bit unsigned integer.

Return Value: A 32-bit unsigned integer equivalent to the number in *value* . A **System.String** containing a number. The base of the number in *value*, which must be 2, 8, 10, or 16.

ToUInt64

[C#] public static ulong ToUInt64(bool value);

[C++] public: static unsigned __int64 ToUInt64(bool value);

[VB] Public Shared Function ToUInt64(ByVal value As Boolean) As UInt64

[JScript] public static function ToUInt64(value : Boolean) : UInt64;

Description

Converts the value of the specified Boolean value to the equivalent 64-bit unsigned integer.

Return Value: The number 1 if *value* is **true** ; otherwise, 0. A Boolean value.

ToUInt64

[C#] public static ulong ToUInt64(byte value);

[C++] public: static unsigned __int64 ToUInt64(unsigned char value);

[VB] Public Shared Function ToUInt64(ByVal value As Byte) As UInt64

1 [JScript] public static function ToUInt64(value : Byte) : UInt64;

3 *Description*

4 Converts the value of the specified 8-bit unsigned integer to the equivalent
5 64-bit signed integer.

6 *Return Value:* The 64-bit signed integer equivalent to the value of *value* . An 8-bit
7 unsigned integer.

8 ToUInt64

10 [C#] public static ulong ToUInt64(char value);

11 [C++] public: static unsigned __int64 ToUInt64(__wchar_t value);

12 [VB] Public Shared Function ToUInt64(ByVal value As Char) As UInt64

13 [JScript] public static function ToUInt64(value : Char) : UInt64;

15 *Description*

16 Converts the value of the specified Unicode character to the equivalent 64-
17 bit unsigned integer.

18 *Return Value:* The 64-bit unsigned integer equivalent to *value* . A Unicode
19 character.

20 ToUInt64

22 [C#] public static ulong ToUInt64(DateTime value);

23 [C++] public: static unsigned __int64 ToUInt64(DateTime value);

24 [VB] Public Shared Function ToUInt64(ByVal value As DateTime) As UInt64

25 [JScript] public static function ToUInt64(value : DateTime) : UInt64;

1
2 *Description*

3 Calling this method always throws **System.InvalidCastException** .

4 This method is reserved for future use. A **System.DateTime**.

5 ToUInt64

6
7 [C#] public static ulong ToUInt64(decimal value);

8 [C++] public: static unsigned __int64 ToUInt64(Decimal value);

9 [VB] Public Shared Function ToUInt64(ByVal value As Decimal) As UInt64

10 [JScript] public static function ToUInt64(value : Decimal) : UInt64;

11
12 *Description*

13 Converts the value of the specified **Decimal** number to an equivalent 64-bit
14 unsigned integer.

15 *Return Value:* *value* rounded to the nearest 64-bit unsigned integer. If *value* is
16 halfway between two whole numbers, the even number is returned; that is, 4.5 is
17 converted to 4, and 5.5 is converted to 6. A **System.Decimal** number.

18 ToUInt64

19
20 [C#] public static ulong ToUInt64(double value);

21 [C++] public: static unsigned __int64 ToUInt64(double value);

22 [VB] Public Shared Function ToUInt64(ByVal value As Double) As UInt64

23 [JScript] public static function ToUInt64(value : double) : UInt64;

24
25 *Description*

Converts the value of the specified double-precision floating point number to an equivalent 64-bit unsigned integer.

Return Value: *value* rounded to the nearest 64-bit unsigned integer. If *value* is halfway between two whole numbers, the even number is returned; that is, 4.5 is converted to 4, and 5.5 is converted to 6. A double-precision floating point number.

ToUInt64

[C#] public static ulong ToUInt64(short value);

[C++] public: static unsigned __int64 ToUInt64(short value);

[VB] Public Shared Function ToUInt64(ByVal value As Short) As UInt64

[JScript] public static function ToUInt64(value : Int16) : UInt64;

Description

Converts the value of the specified 16-bit signed integer to the equivalent 64-bit unsigned integer.

Return Value: The 64-bit unsigned integer equivalent to the value of *value* . A 64-bit signed integer.

ToUInt64

[C#] public static ulong ToUInt64(int value);

[C++] public: static unsigned __int64 ToUInt64(int value);

[VB] Public Shared Function ToUInt64(ByVal value As Integer) As UInt64

[JScript] public static function ToUInt64(value : int) : UInt64;

1
2 *Description*

3 Converts the value of the specified 32-bit signed integer to an equivalent
4 64-bit unsigned integer.

5 *Return Value:* The 64-bit unsigned integer equivalent of *value* . A 32-bit signed
6 integer.

7 ToUInt64

8
9 [C#] public static ulong ToUInt64(long value);

10 [C++] public: static unsigned __int64 ToUInt64(__int64 value);

11 [VB] Public Shared Function ToUInt64(ByVal value As Long) As UInt64

12 [JScript] public static function ToUInt64(value : long) : UInt64;

13
14 *Description*

15 Converts the value of the specified 64-bit signed integer to an equivalent
16 64-bit unsigned integer.

17 *Return Value:* A 64-bit unsigned integer equivalent to the value of *value* . A 64-bit
18 signed integer.

19 ToUInt64

20
21 [C#] public static ulong ToUInt64(object value);

22 [C++] public: static unsigned __int64 ToUInt64(Object* value);

23 [VB] Public Shared Function ToUInt64(ByVal value As Object) As UInt64

24 [JScript] public static function ToUInt64(value : Object) : UInt64; Converts a
25 specified value to a 64-bit unsigned integer.

Description

Converts the value of the specified **Object** to a 64-bit unsigned integer.

Return Value: A 64-bit unsigned integer equivalent to the value of *value* , or zero if *value* is **null** .

The return value is the result of invoking the **ICconvertible.ToInt64** method of the underlying type of *value* . An **System.Object** that implements the **System.IConvertible** interface or **null**.

ToUInt64

[C#] public static ulong ToUInt64(sbyte value);

[C++] public: static unsigned __int64 ToUInt64(char value);

[VB] Public Shared Function ToUInt64(ByVal value As SByte) As UInt64

[JScript] public static function ToUInt64(value : SByte) : UInt64;

Description

Converts the value of the specified 8-bit signed integer to the equivalent 64-bit unsigned integer.

Return Value: The 8-bit unsigned integer equivalent to the value of *value* . An 8-bit signed integer.

ToUInt64

[C#] public static ulong ToUInt64(float value);

[C++] public: static unsigned __int64 ToUInt64(float value);

[VB] Public Shared Function ToUInt64(ByVal value As Single) As UInt64

1 [JScript] public static function ToUInt64(value : float) : UInt64;

3 *Description*

4 Converts the value of the specified single-precision floating point number
5 to an equivalent 64-bit unsigned integer.

6 *Return Value:* *value* rounded to the nearest 64-bit unsigned integer. If *value* is
7 halfway between two whole numbers, the even number is returned; that is, 4.5 is
8 converted to 4, and 5.5 is converted to 6. A single-precision floating point number.

9 ToUInt64

11 [C#] public static ulong ToUInt64(string value);

12 [C++] public: static unsigned __int64 ToUInt64(String* value);

13 [VB] Public Shared Function ToUInt64(ByVal value As String) As UInt64

14 [JScript] public static function ToUInt64(value : String) : UInt64;

16 *Description*

17 Converts the specified **String** representation of a number to an equivalent
18 64-bit signed integer.

19 *Return Value:* A 64-bit signed integer equivalent to the value of *value* .

20 The return value is the result of invoking the
21 **System.Int64.Parse(System.String)** method on *value* . A **System.String**
22 containing a number to convert.

23 ToUInt64

25 [C#] public static ulong ToUInt64(ushort value);

```
1 [C++] public: static unsigned __int64 ToUInt64(unsigned short value);
2 [VB] Public Shared Function ToUInt64(ByVal value As UInt16) As UInt64
3 [JScript] public static function ToUInt64(value : UInt16) : UInt64;
```

Description

Converts the value of the specified 16-bit unsigned integer to the equivalent 64-bit unsigned integer.

Return Value: The 64-bit unsigned integer equivalent to the value of *value* . A 16-bit unsigned integer.

ToUInt64

```
12 [C#] public static ulong ToUInt64(uint value);
13 [C++] public: static unsigned __int64 ToUInt64(unsigned int value);
14 [VB] Public Shared Function ToUInt64(ByVal value As UInt32) As UInt64
15 [JScript] public static function ToUInt64(value : UInt32) : UInt64;
```

Description

Converts the value of the specified 32-bit unsigned integer to an equivalent 64-bit unsigned integer.

Return Value: The 64-bit unsigned integer equivalent of *value* . A 32-bit unsigned integer.

ToUInt64

```
24 [C#] public static ulong ToUInt64(ulong value);
25 [C++] public: static unsigned __int64 ToUInt64(unsigned __int64 value);
```

1 [VB] Public Shared Function ToUInt64(ByVal value As UInt64) As UInt64

2 [JScript] public static function ToUInt64(value : UInt64) : UInt64;

3
4 *Description*

5 Returns the specified 64-bit unsigned integer; no actual conversion is
6 performed.

7 *Return Value:* *value* is returned unchanged. A 64-bit unsigned integer.

8 ToUInt64

9
10 [C#] public static ulong ToUInt64(object value, IFormatProvider provider);

11 [C++] public: static unsigned __int64 ToUInt64(Object* value, IFormatProvider*
12 provider);

13 [VB] Public Shared Function ToUInt64(ByVal value As Object, ByVal provider
14 As IFormatProvider) As UInt64

15 [JScript] public static function ToUInt64(value : Object, provider :
16 IFormatProvider) : UInt64;

17
18 *Description*

19 Converts the value of the specified **Object** to a 64-bit unsigned integer
20 using the specified culture-specific formatting information.

21 *Return Value:* A 64-bit unsigned integer equivalent to the value of *value* , or zero
22 if *value* is **null** .

23 The return value is the result of invoking the **IConvertible.ToUInt64**
24 method of the underlying type of *value* . An **System.Object** that implements the

System.IConvertible interface. An **System.IFormatProvider** interface implementation that supplies culture-specific formatting information.

ToUInt64

[C#] public static ulong ToUInt64(string value, IFormatProvider provider);

[C++] public: static unsigned __int64 ToUInt64(String* value, IFormatProvider* provider);

[VB] Public Shared Function ToUInt64(ByVal value As String, ByVal provider As IFormatProvider) As UInt64

[JScript] public static function ToUInt64(value : String, provider : IFormatProvider) : UInt64;

Description

Converts the specified **String** representation of a number to an equivalent 64-bit unsigned integer using the specified culture-specific formatting information.

Return Value: A 64-bit unsigned integer equivalent to the value of *value* .

The return value is the result of invoking **System.UInt64.Parse(System.String)** on *value* . A **System.String** containing a number to convert. An **System.IFormatProvider** interface implementation that supplies culture-specific formatting information.

ToUInt64

[C#] public static ulong ToUInt64(string value, int fromBase);

[C++] public: static unsigned __int64 ToUInt64(String* value, int fromBase);

[VB] Public Shared Function ToUInt64(ByVal value As String, ByVal fromBase

As Integer) As UInt64

[JScript] public static function ToUInt64(value : String, fromBase : int) : UInt64;

Description

Converts the **String** representation of a number in a specified base to an equivalent 64-bit unsigned integer.

Return Value: A 64-bit unsigned integer equivalent to the number in *value* . A **System.String** containing a number. The base of the number in *value*, which must be 2, 8, 10, or 16.

CrossAppDomainDelegate delegate (System)

ToUInt64

Description

Used by

System.AppDomain.DoCallback(System.CrossAppDomainDelegate) for cross-application domain calls.

Every derived class of **System.Delegate** and **System.MulticastDelegate** has a constructor and an **Invoke** method. See the Managed Extensions for C++ code example given in the description for **System.Delegate** .

DateTime structure (System)

ToUInt64

Description

Represents an instant in time, typically expressed as a date and time of day.

The **DateTime** value type represents dates and times with values ranging from 12:00:00 AM, 1/1/0001 CE (Common Era) to 11:59:59 PM, 12/31/9999 CE.

ToUInt64

[C#] public static readonly DateTime MaxValue;

[C++] public: static DateTime MaxValue;

[VB] Public Shared ReadOnly MaxValue As DateTime

[JScript] public static var MaxValue : DateTime;

Description

A constant representing the largest possible value of **DateTime** .

The value of this constant is 11:59:59 PM, 12/31/9999 CE.

ToUInt64

[C#] public static readonly DateTime MinValue;

[C++] public: static DateTime MinValue;

[VB] Public Shared ReadOnly MinValue As DateTime

[JScript] public static var MinValue : DateTime;

Description

A constant representing the smallest possible value of **DateTime** .

The value of this constant is 12:00:00 AM, 1/1/0001 CE.

DateTime

Example Syntax:

ToUInt64

[C#] public DateTime(long ticks);

[C++] public: DateTime(__int64 ticks);

[VB] Public Sub New(ByVal ticks As Long)

[JScript] public function DateTime(ticks : long); Initializes a new instance of the **DateTime** structure.

Description

Initializes a new instance of the **DateTime** structure to a specified number of ticks. A date and time expressed in 100-nanosecond units.

DateTime

Example Syntax:

ToUInt64

[C#] public DateTime(int year, int month, int day);

[C++] public: DateTime(int year, int month, int day);

[VB] Public Sub New(ByVal year As Integer, ByVal month As Integer, ByVal day As Integer)

[JScript] public function DateTime(year : int, month : int, day : int);

Description

Initializes a new instance of the **DateTime** structure to the specified year, month, and day.

The time of day for the resulting **DateTime** is midnight. The year (1 through 9999). The month (1 through 12). The day (1 through the number of days in *month*).

DateTime

Example Syntax:

ToUInt64

[C#] public DateTime(int year, int month, int day, Calendar calendar);

[C++] public: DateTime(int year, int month, int day, Calendar* calendar);

[VB] Public Sub New(ByVal year As Integer, ByVal month As Integer, ByVal day As Integer, ByVal calendar As Calendar)

[JScript] public function DateTime(year : int, month : int, day : int, calendar : Calendar);

Description

Initializes a new instance of the **DateTime** structure to the specified year, month, and day for the specified calendar.

The time of day for the resulting **DateTime** is midnight. The year (1 through 9999). The month (1 through the number of months in *calendar*). The day (1 through the number of days in *month*). The calendar which applies to this

DateTime.

DateTime

Example Syntax:

ToUInt64

```

1 [C#] public DateTime(int year, int month, int day, int hour, int minute, int second);
2
3 [C++] public: DateTime(int year, int month, int day, int hour, int minute, int
4 second);
5
6 [VB] Public Sub New(ByVal year As Integer, ByVal month As Integer, ByVal
7 day As Integer, ByVal hour As Integer, ByVal minute As Integer, ByVal second
8 As Integer)
9 [JScript] public function DateTime(year : int, month : int, day : int, hour : int,
10 minute : int, second : int);
11

```

Description

Initializes a new instance of the **DateTime** structure to the specified year, month, day, hour, minute, and second. The year (1 through 9999) The month (1 through 12) The day (1 through the number of days in *month*) The hours (0 through 23) The minutes (0 through 59) The seconds (0 through 59)

DateTime

Example Syntax:

ToUInt64

```

20 [C#] public DateTime(int year, int month, int day, int hour, int minute, int second,
21 Calendar calendar);
22
23 [C++] public: DateTime(int year, int month, int day, int hour, int minute, int
24 second, Calendar* calendar);
25
26 [VB] Public Sub New(ByVal year As Integer, ByVal month As Integer, ByVal
27 day As Integer, ByVal hour As Integer, ByVal minute As Integer, ByVal second

```

As Integer, ByVal calendar As Calendar)

[JScript] public function DateTime(year : int, month : int, day : int, hour : int,
minute : int, second : int, calendar : Calendar);

Description

Initializes a new instance of the **DateTime** structure to the specified year,
month, day, hour, minute, and second for the specified calendar.

The time of day for the resulting **DateTime** is midnight. The year (1
through 9999) The month (1 through the number of months in *calendar*) The day
(1 through the number of days in *month*) The hours (0 through 23) The minutes (0
through 59) The seconds (0 through 59) The calendar which applies to this

DateTime

DateTime

Example Syntax:

ToUInt64

[C#] public DateTime(int year, int month, int day, int hour, int minute, int second,
int millisecond);

[C++] public: DateTime(int year, int month, int day, int hour, int minute, int
second, int millisecond);

[VB] Public Sub New(ByVal year As Integer, ByVal month As Integer, ByVal
day As Integer, ByVal hour As Integer, ByVal minute As Integer, ByVal second
As Integer, ByVal millisecond As Integer)

[JScript] public function DateTime(year : int, month : int, day : int, hour : int,
minute : int, second : int, millisecond : int);

Description

Initializes a new instance of the **DateTime** structure to the specified year, month, day, hour, minute, second, and millisecond. The year (1 through 9999) The month (1 through 12) The day (1 through the number of days in *month*) The hours (0 through 23) The minutes (0 through 59) The seconds (0 through 59) The milliseconds

DateTime

Example Syntax:

ToUInt64

```
[C#] public DateTime(int year, int month, int day, int hour, int minute, int second, int millisecond, Calendar calendar);
```

```
[C++] public: DateTime(int year, int month, int day, int hour, int minute, int second, int millisecond, Calendar* calendar);
```

```
[VB] Public Sub New(ByVal year As Integer, ByVal month As Integer, ByVal day As Integer, ByVal hour As Integer, ByVal minute As Integer, ByVal second As Integer, ByVal millisecond As Integer, ByVal calendar As Calendar)
```

```
[JScript] public function DateTime(year : int, month : int, day : int, hour : int, minute : int, second : int, millisecond : int, calendar : Calendar);
```

Description

Initializes a new instance of the **DateTime** structure to the specified year, month, day, hour, minute, second, and millisecond for the specified calendar.

The time of day for the resulting **DateTime** is midnight. The year (1 through 9999) The month (1 through the number of months in *calendar*) The day (1 through the number of days in *month*) The hours (0 through 23) The minutes (0 through 59) The seconds (0 through 59) The milliseconds The calendar which applies to this **System.DateTime**

Date

ToUInt64

[C#] public DateTime Date {get;}

[C++] public: __property DateTime get_Date();

[VB] Public ReadOnly Property Date As DateTime

[JScript] public function get Date() : DateTime;

Description

Gets the date component of this instance.

Day

ToUInt64

[C#] public int Day {get;}

[C++] public: __property int get_Day();

[VB] Public ReadOnly Property Day As Integer

[JScript] public function get Day() : int;

Description

Gets the day of the month represented by this instance.

1 DayOfWeek
2 ToUInt64
3
4 [C#] public DayOfWeek DayOfWeek {get;}
5 [C++] public: __property DayOfWeek get_DayOfWeek();
6 [VB] Public ReadOnly Property DayOfWeek As DayOfWeek
7 [JScript] public function get DayOfWeek() : DayOfWeek;

8
9 *Description*

10 Gets the day of the week represented by this instance.

11 DayOfYear
12 ToUInt64

13
14 [C#] public int DayOfYear {get;}
15 [C++] public: __property int get_DayOfYear();
16 [VB] Public ReadOnly Property DayOfYear As Integer
17 [JScript] public function get DayOfYear() : int;

18
19 *Description*

20 Gets the day of the year represented by this instance.

21 Hour
22 ToUInt64

23
24 [C#] public int Hour {get;}
25 [C++] public: __property int get_Hour();

1 [VB] Public ReadOnly Property Hour As Integer

2 [JScript] public function get Hour() : int;

3
4 *Description*

5 Gets the hour component of the date represented by this instance.

6 Millisecond

7 ToUInt64

8
9 [C#] public int Millisecond {get;}

10 [C++] public: __property int get_Millisecond();

11 [VB] Public ReadOnly Property Millisecond As Integer

12 [JScript] public function get Millisecond() : int;

13
14 *Description*

15 Gets the milliseconds component of the date represented by this instance.

16 Minute

17 ToUInt64

18
19 [C#] public int Minute {get;}

20 [C++] public: __property int get_Minute();

21 [VB] Public ReadOnly Property Minute As Integer

22 [JScript] public function get Minute() : int;

23
24 *Description*

25 Gets the minute component of the date represented by this instance.

Month

ToUInt64

[C#] public int Month {get;}

[C++] public: __property int get_Month();

[VB] Public ReadOnly Property Month As Integer

[JScript] public function get Month() : int;

Description

Gets the month component of the date represented by this instance.

Now

ToUInt64

[C#] public static DateTime Now {get;}

[C++] public: __property static DateTime get_Now();

[VB] Public Shared ReadOnly Property Now As DateTime

[JScript] public static function get Now() : DateTime;

Description

Gets a **DateTime** that is the current local time on this computer.

The resolution of this property depends on the system timer.

Second

ToUInt64

[C#] public int Second {get;}

```

1 [C++] public: __property int get_Second();
2 [VB] Public ReadOnly Property Second As Integer
3 [JScript] public function get Second() : int;
4

```

Description

Retrieves the seconds component of the date represented by this instance.

Ticks

ToUInt64

```

10 [C#] public long Ticks {get;}
11 [C++] public: __property __int64 get_Ticks();
12 [VB] Public ReadOnly Property Ticks As Long
13 [JScript] public function get Ticks() : long;
14

```

Description

Gets the number of 100-nanosecond ticks that represent the date and time of this instance.

The value of this property is the number of 100-nanosecond intervals that have elapsed since 1/1/0001, 12:00am.

TimeOfDay

ToUInt64

```

23 [C#] public TimeSpan TimeOfDay {get;}
24 [C++] public: __property TimeSpan get_TimeOfDay();
25 [VB] Public ReadOnly Property TimeOfDay As TimeSpan

```

1 [JScript] public function get TimeOfDay() : TimeSpan;

3 *Description*

4 Gets the time of day for this instance.

5 Today

6 ToUInt64

8 [C#] public static DateTime Today {get;}

9 [C++] public: __property static DateTime get_ Today();

10 [VB] Public Shared ReadOnly Property Today As DateTime

11 [JScript] public static function get Today() : DateTime;

13 *Description*

14 Gets the current date.

15 UtcNow

16 ToUInt64

18 [C#] public static DateTime UtcNow {get;}

19 [C++] public: __property static DateTime get_ UtcNow();

20 [VB] Public Shared ReadOnly Property UtcNow As DateTime

21 [JScript] public static function get UtcNow() : DateTime;

23 *Description*

24 Gets a **DateTime** that is the current local time on this computer expressed
25 as the coordinated universal time (UTC).

The resolution of this property depends on the system timer.

Year

ToUInt64

[C#] public int Year {get;}

[C++] public: __property int get_Year();

[VB] Public ReadOnly Property Year As Integer

[JScript] public function get Year() : int;

Description

Gets the year component of the date represented by this instance.

Add

[C#] public DateTime Add(TimeSpan value);

[C++] public: DateTime Add(TimeSpan value);

[VB] Public Function Add(ByVal value As TimeSpan) As DateTime

[JScript] public function Add(value : TimeSpan) : DateTime;

Description

Adds the value of the specified **TimeSpan** instance to the value of this instance.

Return Value: A **DateTime** whose value is the sum of the date and time represented by this instance and the time interval represented by *value*.

This method does not change the value of this **DateTime** instance. Instead, a new **DateTime** instance is returned whose value is the result of this operation. A time interval.

AddDays

[C#] public DateTime AddDays(double value);

[C++] public: DateTime AddDays(double value);

[VB] Public Function AddDays(ByVal value As Double) As DateTime

[JScript] public function AddDays(value : double) : DateTime;

Description

Adds the specified number of days to the value of this instance.

Return Value: A **DateTime** whose value is the sum of the date and time represented by this instance and the number of days represented by *value*.

This method does not change the value of this **DateTime** instance. Instead, a new **DateTime** instance is returned whose value is the result of this operation. A number of whole and fractional days.

AddHours

[C#] public DateTime AddHours(double value);

[C++] public: DateTime AddHours(double value);

[VB] Public Function AddHours(ByVal value As Double) As DateTime

[JScript] public function AddHours(value : double) : DateTime;

Description

Adds the specified number of hours to the value of this instance.

Return Value: A **DateTime** whose value is the sum of the date and time represented by this instance and the number of hours represented by *value* .

This method does not change the value of this **DateTime** instance. Instead, a new **DateTime** instance is returned whose value is the result of this operation. A number of whole and fractional hours.

AddMilliseconds

[C#] public DateTime AddMilliseconds(double value);

[C++] public: DateTime AddMilliseconds(double value);

[VB] Public Function AddMilliseconds(ByVal value As Double) As DateTime

[JScript] public function AddMilliseconds(value : double) : DateTime;

Description

Adds the specified number of milliseconds to the value of this instance.

Return Value: A **DateTime** whose value is the sum of the date and time represented by this instance and the number of milliseconds represented by *value* .

This method does not change the value of this **DateTime** instance. Instead, a new **DateTime** instance is returned whose value is the result of this operation. A number of milliseconds.

AddMinutes

[C#] public DateTime AddMinutes(double value);

[C++] public: DateTime AddMinutes(double value);

[VB] Public Function AddMinutes(ByVal value As Double) As DateTime

1 [JScript] public function AddMinutes(value : double) : DateTime;

3 *Description*

4 Adds the specified number of minutes to the value of this instance.

5 *Return Value:* A **DateTime** whose value is the sum of the date and time
6 represented by this instance and the number of minutes represented by *value* .

7 This method does not change the value of this **DateTime** instance. Instead,
8 a new **DateTime** instance is returned whose value is the result of this operation. A
9 number of whole and fractional minutes.

10 **AddMonths**

12 [C#] public DateTime AddMonths(int months);

13 [C++] public: DateTime AddMonths(int months);

14 [VB] Public Function AddMonths(ByVal months As Integer) As DateTime

15 [JScript] public function AddMonths(months : int) : DateTime;

17 *Description*

18 Adds the specified number of months to the value of this instance.

19 *Return Value:* A **DateTime** whose value is the sum of the date and time
20 represented by this instance and *months* .

21 This method does not change the value of this **DateTime** instance. Instead,
22 a new **DateTime** instance is returned whose value is the result of this operation. A
23 number of months.

24 **AddSeconds**

1
2 [C#] public DateTime AddSeconds(double value);

3 [C++] public: DateTime AddSeconds(double value);

4 [VB] Public Function AddSeconds(ByVal value As Double) As DateTime

5 [JScript] public function AddSeconds(value : double) : DateTime;

6
7 *Description*

8 Adds the specified number of seconds to the value of this instance.

9 *Return Value:* A **DateTime** whose value is the sum of the date and time
10 represented by this instance and the number of seconds represented by *value* .

11 This method does not change the value of this **DateTime** instance. Instead,
12 a new **DateTime** instance is returned whose value is the result of this operation. A
13 number of whole and fractional seconds.

14 *AddTicks*

15
16 [C#] public DateTime AddTicks(long value);

17 [C++] public: DateTime AddTicks(__int64 value);

18 [VB] Public Function AddTicks(ByVal value As Long) As DateTime

19 [JScript] public function AddTicks(value : long) : DateTime;

20
21 *Description*

22 Adds the specified number of ticks to the value of this instance.

23 *Return Value:* A **DateTime** whose value is the sum of the date and time
24 represented by this instance and the time represented by *value* .

This method does not change the value of this **DateTime** instance. Instead, a new **DateTime** instance is returned whose value is the result of this operation. A number of 100-nanosecond ticks.

AddYears

```
[C#] public DateTime AddYears(int value);  
[C++] public: DateTime AddYears(int value);  
[VB] Public Function AddYears(ByVal value As Integer) As DateTime  
[JScript] public function AddYears(value : int) : DateTime;
```

Description

Adds the specified number of years to the value of this instance.

Return Value: A **DateTime** whose value is the sum of the date and time represented by this instance and the number of years represented by *value* .

This method does not change the value of this **DateTime** instance. Instead, a new **DateTime** instance is returned whose value is the result of this operation. A number of years.

Compare

```
[C#] public static int Compare(DateTime t1, DateTime t2);  
[C++] public: static int Compare(DateTime t1, DateTime t2);  
[VB] Public Shared Function Compare(ByVal t1 As DateTime, ByVal t2 As  
DateTime) As Integer  
[JScript] public static function Compare(t1 : DateTime, t2 : DateTime) : int;
```

Description

Compares two instances of **DateTime** and returns an indication of their relative values.

Return Value: A signed number indicating the relative values of *t1* and *t2* . The first **DateTime**. The second **DateTime**.

CompareTo

[C#] public int CompareTo(object value);

[C++] public: __sealed int CompareTo(Object* value);

[VB] NotOverridable Public Function CompareTo(ByVal value As Object) As Integer

[JScript] public function CompareTo(value : Object) : int;

Description

Compares this instance to a specified object and returns an indication of their relative values.

Return Value: A signed number indicating the relative values of this instance and *value* .

Any instance of **DateTime** , regardless of its value, is considered greater than **null** . An object to compare, or **null**.

DaysInMonth

[C#] public static int DaysInMonth(int year, int month);

[C++] public: static int DaysInMonth(int year, int month);

1 [VB] Public Shared Function DaysInMonth(ByVal year As Integer, ByVal month
2 As Integer) As Integer

3 [JScript] public static function DaysInMonth(year : int, month : int) : int;

4
5 *Description*

6 Returns the number of days in the specified month of the specified year.

7 *Return Value:* The number of days in *month* for the specified *year* . The year. The
8 month (a number ranging from 1 to 12).

9 *Equals*

10
11 [C#] public override bool Equals(object value);

12 [C++] public: bool Equals(Object* value);

13 [VB] Overrides Public Function Equals(ByVal value As Object) As Boolean

14 [JScript] public override function Equals(value : Object) : Boolean; Returns a
15 value indicating whether an instance of **DateTime** is equal to a specified object.

16
17 *Description*

18 Returns a value indicating whether this instance is equal to a specified
19 object.

20 *Return Value:* **true** if *value* is an instance of **DateTime** and equals the value of
21 this instance; otherwise, **false** . An object to compare with this instance.

22 *Equals*

23
24 [C#] public static new bool Equals(DateTime t1, DateTime t2);

25 [C++] public: static bool Equals(DateTime t1, DateTime t2);

1 [VB] Shadows Public Shared Function Equals(ByVal t1 As DateTime, ByVal t2
2 As DateTime) As Boolean

3 [JScript] public static hide function Equals(t1 : DateTime, t2 : DateTime) :
4 Boolean;

5
6 *Description*

7 Returns a value indicating whether two instances of **System.DateTime** are
8 equal.

9 *Return Value:* **true** if the two **DateTime** values are equal; otherwise, **false** . The
10 first **DateTime**. The second **DateTime**.

11 *FromFileTime*

12
13 [C#] public static DateTime FromFileTime(long fileTime);

14 [C++] public: static DateTime FromFileTime(__int64 fileTime);

15 [VB] Public Shared Function FromFileTime(ByVal fileTime As Long) As
16 DateTime

17 [JScript] public static function FromFileTime(fileTime : long) : DateTime;

18
19 *Description*

20 Returns a **DateTime** equivalent to the specified operating system file
21 timestamp.

22 *Return Value:* A **DateTime** value representing the date and time of *fileTime* ,
23 adjusted to local time.

24 *fileTime* is a 64-bit signed integer value representing a Windows file
25 timestamp. The timestamp is the number of 100-nanosecond intervals that have

elapsed since 1/1/1601 12:00 AM coordinated universal time (UTC). A Windows file time.

FromOADate

[C#] public static DateTime FromOADate(double d);

[C++] public: static DateTime FromOADate(double d);

[VB] Public Shared Function FromOADate(ByVal d As Double) As DateTime

[JScript] public static function FromOADate(d : double) : DateTime;

Description

Returns a **DateTime** equivalent to the specified OLE Automation Date.

Return Value: A **DateTime** that represents the same date and time as *d*.

d must be a value between negative 657435.0 through positive 2958466.0.

It is stored as a double-precision floating point number. An OLE Automation Date value.

GetDateTimeFormats

[C#] public string[] GetDateTimeFormats();

[C++] public: String* GetDateTimeFormats() __gc[];

[VB] Public Function GetDateTimeFormats() As String()

[JScript] public function GetDateTimeFormats() : String[]; Converts the value of this instance to all of the **String** representations supported by the standard **DateTime** format specifiers.

Description

Converts the value of this instance to all of the **String** representations supported by the standard **DateTime** format specifiers.

Return Value: A **String** array where each element is the representation of the value of this instance formatted with one of the standard **DateTime** formatting specifiers.

Each element of the return value is formatted using information from the current culture. For more information about culture-specific formatting information for the current culture, see **System.Globalization.CultureInfo.CurrentCulture**.

GetDateTimeFormats

[C#] public string[] GetDateTimeFormats(char format);

[C++] public: String* GetDateTimeFormats(__wchar_t format) __gc[];

[VB] Public Function GetDateTimeFormats(ByVal format As Char) As String()

[JScript] public function GetDateTimeFormats(format : Char) : String[];

Description

Converts the value of this instance to all of the **String** representations supported by the specified standard **DateTime** format specifier.

Return Value: A **String** array where each element is the representation of the value of this instance formatted with the *format* standard **DateTime** formatting specifier.

Each element of the return value is formatted using information from the current culture. For more information about culture-specific formatting information for the current culture, see

System.Globalization.CultureInfo.CurrentCulture . A Unicode character containing a format specifier.

GetDateTimeFormats

```
[C#] public string[] GetDateTimeFormats(IFormatProvider provider);  
[C++] public: String* GetDateTimeFormats(IFormatProvider* provider) __gc[];  
[VB] Public Function GetDateTimeFormats(ByVal provider As IFormatProvider)  
As String()  
[JScript] public function GetDateTimeFormats(provider : IFormatProvider) :  
String[];
```

Description

Converts the value of this instance to all of the **String** representations supported by the standard **DateTime** format specifiers and the specified culture-specific formatting information.

Return Value: A **String** array where each element is the representation of the value of this instance formatted with one of the standard **DateTime** formatting specifiers.

Each element of the return value is formatted using culture-specific information supplied by *provider* . An **System.IFormatProvider** interface implementation that supplies culture-specific formatting information about this instance.

GetDateTimeFormats

```
[C#] public string[] GetDateTimeFormats(char format, IFormatProvider provider);
```

```

1 [C++] public: String* GetDateTimeFormats(__wchar_t format, IFormatProvider*
2 provider) __gc[];
3 [VB] Public Function GetDateTimeFormats(ByVal format As Char, ByVal
4 provider As IFormatProvider) As String()
5 [JScript] public function GetDateTimeFormats(format : Char, provider :
6 IFormatProvider) : String[];
7

```

8 *Description*

9 Converts the value of this instance to all of the **String** representations
10 supported by the specified standard **DateTime** format specifier and culture-
11 specific formatting information.

12 *Return Value:* A **String** array where each element is the representation of the
13 value of this instance formatted with one of the standard **DateTime** formatting
14 specifiers.

15 Each element of the return value is formatted using culture-specific
16 information supplied by *provider* . A Unicode character containing a format
17 specifier. An **System.IFormatProvider** interface implementation that supplies
18 culture-specific formatting information about this instance.

19 *GetHashCode*

```

20
21 [C#] public override int GetHashCode();
22 [C++] public: int GetHashCode();
23 [VB] Overrides Public Function GetHashCode() As Integer
24 [JScript] public override function GetHashCode() : int;
25

```


1
2 *Description*

3 Returns the hash code for this instance.

4 *Return Value:* A 32-bit signed integer hash code.

5 GetTypeCode

6
7 [C#] public TypeCode GetTypeCode();

8 [C++] public: __sealed TypeCode GetTypeCode();

9 [VB] NotOverridable Public Function GetTypeCode() As TypeCode

10 [JScript] public function GetTypeCode() : TypeCode;

11
12 *Description*

13 Returns the **TypeCode** for value type **DateTime** .

14 *Return Value:* The enumerated constant, **System.TypeCode.DateTime** .

15 IsLeapYear

16
17 [C#] public static bool IsLeapYear(int year);

18 [C++] public: static bool IsLeapYear(int year);

19 [VB] Public Shared Function IsLeapYear(ByVal year As Integer) As Boolean

20 [JScript] public static function IsLeapYear(year : int) : Boolean;

21
22 *Description*

23 Returns an indication whether the specified year is a leap year.

24 *Return Value:* **true** if the year is a leap year; otherwise, **false** .

1 *year* is specified as a 4-digit base 10 number; for example, 1996. A 4-digit
2 *year*.

3 *op_Addition*

4
5 [C#] public static DateTime operator +(DateTime d, TimeSpan t);

6 [C++] public: static DateTime op_Addition(DateTime d, TimeSpan t);

7 [VB] returnValue = DateTime.op_Addition(d, t)

8 [JScript] returnValue = d + t;

9
10 *Description*

11 Adds a specified time interval to a specified date and time, yielding a new
12 date and time.

13 *Return Value:* A **DateTime** that is the sum of the values of *d* and *t* . A date and
14 time. A time interval.

15 *op_Equality*

16
17 [C#] public static bool operator ==(DateTime d1, DateTime d2);

18 [C++] public: static bool op_Equality(DateTime d1, DateTime d2);

19 [VB] returnValue = DateTime.op_Equality(d1, d2)

20 [JScript] returnValue = d1 == d2;

21
22 *Description*

23 Determines whether two specified instances of **DateTime** are equal.

24 *Return Value:* **true** if *d1* and *d2* represent the same date and time; otherwise, **false**
25 . A **DateTime**. A **DateTime**.

op_GreaterThan

[C#] public static bool operator >(DateTime t1, DateTime t2);

[C++] public: static bool op_GreaterThan(DateTime t1, DateTime t2);

[VB] returnValue = DateTime.op_GreaterThan(t1, t2)

[JScript] returnValue = t1 > t2;

Description

Determines whether one specified **DateTime** is greater than another specified **DateTime** .

Return Value: **true** if *t1* is greater than *t2* ; otherwise, **false** . A **DateTime**. A **DateTime**.

op_GreaterThanOrEqual

[C#] public static bool operator >=(DateTime t1, DateTime t2);

[C++] public: static bool op_GreaterThanOrEqual(DateTime t1, DateTime t2);

[VB] returnValue = DateTime.op_GreaterThanOrEqual(t1, t2)

[JScript] returnValue = t1 >= t2;

Description

Determines whether one specified **DateTime** is greater than or equal to another specified **DateTime** .

Return Value: **true** if *t1* is greater than or equal to *t2* ; otherwise, **false** . A **DateTime**. A **DateTime**.

op_Inequality

```

1
2 [C#] public static bool operator !=(DateTime d1, DateTime d2);
3 [C++] public: static bool op_Inequality(DateTime d1, DateTime d2);
4 [VB] returnValue = DateTime.op_Inequality(d1, d2)
5 [JScript] returnValue = d1 != d2;
6

```

Description

Determines whether two specified instances of **DateTime** are not equal.

Return Value: **true** if *d1* and *d2* do not represent the same date and time; otherwise, **false** . A **DateTime**. A **DateTime**.

op_LessThan

```

13 [C#] public static bool operator
14 [C++] public: static bool op_LessThan(DateTime t1, DateTime t2);
15 [VB] returnValue = DateTime.op_LessThan(t1, t2)
16 [JScript] returnValue = t1 < t2;
17

```

Description

Determines whether one specified **DateTime** is less than another specified **DateTime** .

Return Value: **true** if *t1* is less than *t2* ; otherwise, **false** . A **DateTime**. A **DateTime**.

op_LessThanOrEqual

```

25 [C#] public static bool operator <=(DateTime t1, DateTime t2);

```

1 [C++] public: static bool op_LessThanOrEqual(DateTime t1, DateTime t2);

2 [VB] returnValue = DateTime.op_LessThanOrEqual(t1, t2)

3 [JScript] returnValue = t1 <= t2;

4
5 *Description*

6 Determines whether one specified **DateTime** is less than or equal to
7 another specified **DateTime** .

8 *Return Value:* **true** if *t1* is less than or equal to *t2* ; otherwise, **false** . A **DateTime**.

9 A **DateTime**.

10 op_Subtraction

11
12 [C#] public static TimeSpan operator -(DateTime d1, DateTime d2);

13 [C++] public: static TimeSpan op_Subtraction(DateTime d1, DateTime d2);

14 [VB] returnValue = DateTime.op_Subtraction(d1, d2)

15 [JScript] returnValue = d1 - d2;

16
17 *Description*

18 Subtracts a specified date and time from another specified date and time,
19 yielding a time interval.

20 *Return Value:* A **System.TimeSpan** that is the time interval between *d1* and *d2* ;
21 that is, *d1* minus *d2* . A **DateTime** (the minuend). A **DateTime** (the subtrahend).

22 op_Subtraction

23
24 [C#] public static DateTime operator -(DateTime d, TimeSpan t);

25 [C++] public: static DateTime op_Subtraction(DateTime d, TimeSpan t);

1 [VB] returnValue = DateTime.op_Subtraction(d, t)

2 [JScript] returnValue = d - t; Subtracts a specified **System.DateTime** or
3 **System.TimeSpan** instance from a specified **System.DateTime** instance.

4
5 *Description*

6 Subtracts a specified time interval from a specified date and time, yielding a
7 new date and time.

8 *Return Value:* A **DateTime** whose value is the value of *d* minus the value of *t* .

9 This method subtracts the ticks values of *t* from the ticks value of *d* . A
10 **DateTime**. A **System.TimeSpan**.

11 *Parse*

12
13 [C#] public static DateTime Parse(string s);

14 [C++] public: static DateTime Parse(String* s);

15 [VB] Public Shared Function Parse(ByVal s As String) As DateTime

16 [JScript] public static function Parse(s : String) : DateTime; Converts the specified
17 **String** representation of a date and time to its **DateTime** equivalent.

18
19 *Description*

20 Converts the specified **String** representation of a date and time to its
21 **DateTime** equivalent.

22 *Return Value:* A **DateTime** equivalent to the date and time contained in *s* .

23 This method attempts to parse *s* completely and avoid throwing
24 **FormatException** . It ignores unrecognized data if possible and fills in missing

month, day, and year information with the current time. A **System.String** containing a date and time to convert.

Parse

[C#] public static DateTime Parse(string s, IFormatProvider provider);

[C++] public: static DateTime Parse(String* s, IFormatProvider* provider);

[VB] Public Shared Function Parse(ByVal s As String, ByVal provider As IFormatProvider) As DateTime

[JScript] public static function Parse(s : String, provider : IFormatProvider) : DateTime;

Description

Converts the specified **String** representation of a date and time to its **DateTime** equivalent using the specified culture-specific format information.

Return Value: A **DateTime** equivalent to the date and time contained in *s* as specified by *provider*.

This method attempts to parse *s* completely and avoid throwing **FormatException**. It ignores unrecognized data if possible and fills in missing month, day, and year information with the current time. A **System.String** containing a date and time to convert. An **System.IFormatProvider** object that supplies culture-specific format information about *s*.

Parse

[C#] public static DateTime Parse(string s, IFormatProvider provider, DateTimeStyles styles);

```

1 [C++] public: static DateTime Parse(String* s, IFormatProvider* provider,
2 DateTimeStyles styles);
3 [VB] Public Shared Function Parse(ByVal s As String, ByVal provider As
4 IFormatProvider, ByVal styles As DateTimeStyles) As DateTime
5 [JScript] public static function Parse(s : String, provider : IFormatProvider, styles :
6 DateTimeStyles) : DateTime;

```

Description

Converts the specified **String** representation of a date and time to its **DateTime** equivalent using the specified culture-specific format information and formatting style.

Return Value: A **DateTime** equivalent to the date and time contained in *s* as specified by *provider* and *styles* .

This method attempts to parse *s* completely and avoid throwing **FormatException** . It ignores unrecognized data if possible and fills in missing month, day, and year information with the current time. A **System.String** containing a date and time to convert. An **System.IFormatProvider** interface implementation that supplies culture-specific formatting information about *s*. The combination of one or more **System.Globalization.DateTimeStyles** constants that indicate the permitted format of *s*.

ParseExact

```

23 [C#] public static DateTime ParseExact(string s, string format, IFormatProvider
24 provider);
25 [C++] public: static DateTime ParseExact(String* s, String* format,

```


1 IFormatProvider* provider);

2 [VB] Public Shared Function ParseExact(ByVal s As String, ByVal format As
3 String, ByVal provider As IFormatProvider) As DateTime

4 [JScript] public static function ParseExact(s : String, format : String, provider :
5 IFormatProvider) : DateTime; Converts the specified **String** representation of a
6 date and time to its **DateTime** equivalent. The format of the **String** representation
7 must match a specified format exactly.

9 *Description*

10 Converts the specified **String** representation of a date and time to its
11 **DateTime** equivalent using the specified format and culture-specific format
12 information. The format of the **String** representation must match the specified
13 format exactly.

14 *Return Value:* A **DateTime** equivalent to the date and time contained in *s* as
15 specified by *format* and *provider* .

16 This method throws **FormatException** if the format of *s* is not exactly as
17 specified by the format pattern in *format* . If *format* consists of a single standard
18 format character, the format pattern that character represents is used. For more
19 information, see the **System.Globalization.DateTimeFormatInfo** topic. A
20 **System.String** containing a date and time to convert. The expected format of *s*.
21 An **System.IFormatProvider** object that supplies culture-specific format
22 information about *s*.

23 ParseExact

24
25 [C#] public static DateTime ParseExact(string s, string format, IFormatProvider

```

1 provider, DateTimeStyles style);
2 [C++] public: static DateTime ParseExact(String* s, String* format,
3 IFormatProvider* provider, DateTimeStyles style);
4 [VB] Public Shared Function ParseExact(ByVal s As String, ByVal format As
5 String, ByVal provider As IFormatProvider, ByVal style As DateTimeStyles) As
6 DateTime
7 [JScript] public static function ParseExact(s : String, format : String, provider :
8 IFormatProvider, style : DateTimeStyles) : DateTime;

```

Description

Converts the specified **String** representation of a date and time to its **DateTime** equivalent using the specified format, culture-specific format information, and style. The format of the **String** representation must match the specified format exactly.

Return Value: A **DateTime** equivalent to the date and time contained in *s* as specified by *format* , *provider* , and *style* .

This method throws **FormatException** if the format of *s* is not exactly as specified by the format pattern in *format* . If *format* consists of a single standard format character, the format pattern that character represents is used. For more information, see the **System.Globalization.DateTimeFormatInfo** topic. A **System.String** containing a date and time to convert. The expected format of *s* . An **System.IFormatProvider** interface implementation that supplies culture-specific formatting information about *s* . The combination of one or more **System.Globalization.DateTimeStyles** constants that indicate the permitted format of *s* .

ParseExact

```
[C#] public static DateTime ParseExact(string s, string[] formats, IFormatProvider provider, DateTimeStyles style);
```

```
[C++] public: static DateTime ParseExact(String* s, String* formats __gc[], IFormatProvider* provider, DateTimeStyles style);
```

```
[VB] Public Shared Function ParseExact(ByVal s As String, ByVal formats() As String, ByVal provider As IFormatProvider, ByVal style As DateTimeStyles) As DateTime
```

```
[JScript] public static function ParseExact(s : String, formats : String[], provider : IFormatProvider, style : DateTimeStyles) : DateTime;
```

Description

Converts the specified **String** representation of a date and time to its **DateTime** equivalent using the specified array of formats, culture-specific format information, and style. The format of the **String** representation must match at least one of the specified formats exactly.

Return Value: A **DateTime** equivalent to the date and time contained in *s* as specified by *formats* , *provider* , and *style* .

This method throws **FormatException** if the format of *s* is not exactly as specified by at least one of the format patterns in *formats* . If an element of *formats* consists of a single standard format character, the format pattern that character represents is used. For more information, see the **System.Globalization.DateTimeFormatInfo** topic. A **System.String** containing one or more dates and times to convert. An array of expected formats of *s* . An

System.IFormatProvider object that supplies culture-specific format information about *s*. The combination of one or more

System.Globalization.DateTimeStyles constants that indicate the permitted format of *s*.

Subtract

[C#] public TimeSpan Subtract(DateTime value);

[C++] public: TimeSpan Subtract(DateTime value);

[VB] Public Function Subtract(ByVal value As DateTime) As TimeSpan

[JScript] public function Subtract(value : DateTime) : TimeSpan; Subtracts the specified time or duration from this instance.

Description

Subtracts the specified date and time from this instance.

Return Value: A **System.TimeSpan** interval equal to the date and time represented by this instance minus the date and time represented by *value*.

This method does not change the value of this **DateTime** instance. Instead, a new **TimeSpan** instance is returned whose value is the result of this operation. A instance of **DateTime**.

Subtract

[C#] public DateTime Subtract(TimeSpan value);

[C++] public: DateTime Subtract(TimeSpan value);

[VB] Public Function Subtract(ByVal value As TimeSpan) As DateTime

[JScript] public function Subtract(value : TimeSpan) : DateTime;

Description

Subtracts the specified **TimeSpan** from this instance.

Return Value: A new **DateTime** equal to the date and time represented by this instance minus time interval, *value* .

This method does not change the value of this **DateTime** instance. Instead, a new **DateTime** instance is returned whose value is the result of this operation.

An instance of **System.TimeSpan**.

IConvertible.ToBoolean

[C#] bool IConvertible.ToBoolean(IFormatProvider provider);

[C++] bool IConvertible::ToBoolean(IFormatProvider* provider);

[VB] Function ToBoolean(ByVal provider As IFormatProvider) As Boolean

Implements IConvertible.ToBoolean

[JScript] function IConvertible.ToBoolean(provider : IFormatProvider) : Boolean;

IConvertible.ToByte

[C#] byte IConvertible.ToByte(IFormatProvider provider);

[C++] unsigned char IConvertible::ToByte(IFormatProvider* provider);

[VB] Function ToByte(ByVal provider As IFormatProvider) As Byte Implements

IConvertible.ToByte

[JScript] function IConvertible.ToByte(provider : IFormatProvider) : Byte;

IConvertible.ToChar

[C#] char IConvertible.ToChar(IFormatProvider provider);

```

1  [C++] __wchar_t IConvertible::ToChar(IFormatProvider* provider);
2  [VB] Function ToChar(ByVal provider As IFormatProvider) As Char Implements
3  IConvertible.ToChar
4  [JScript] function IConvertible.ToChar(provider : IFormatProvider) : Char;
5      IConvertible.ToDateTime
6
7  [C#] DateTime IConvertible.ToDateTime(IFormatProvider provider);
8  [C++] DateTime IConvertible::ToDateTime(IFormatProvider* provider);
9  [VB] Function ToDateTime(ByVal provider As IFormatProvider) As DateTime
10 Implements IConvertible.ToDateTime
11 [JScript] function IConvertible.ToDateTime(provider : IFormatProvider) :
12 DateTime;
13      IConvertible.ToDecimal
14
15 [C#] decimal IConvertible.ToDecimal(IFormatProvider provider);
16 [C++] Decimal IConvertible::ToDecimal(IFormatProvider* provider);
17 [VB] Function ToDecimal(ByVal provider As IFormatProvider) As Decimal
18 Implements IConvertible.ToDecimal
19 [JScript] function IConvertible.ToDecimal(provider : IFormatProvider) : Decimal;
20      IConvertible.ToDouble
21
22 [C#] double IConvertible.ToDouble(IFormatProvider provider);
23 [C++] double IConvertible::ToDouble(IFormatProvider* provider);
24 [VB] Function ToDouble(ByVal provider As IFormatProvider) As Double
25

```

1 Implements IConvertible.ToDouble

2 [JScript] function IConvertible.ToDouble(provider : IFormatProvider) : double;

3 IConvertible.ToInt16

5 [C#] short IConvertible.ToInt16(IFormatProvider provider);

6 [C++] short IConvertible::ToInt16(IFormatProvider* provider);

7 [VB] Function ToInt16(ByVal provider As IFormatProvider) As Short

8 Implements IConvertible.ToInt16

9 [JScript] function IConvertible.ToInt16(provider : IFormatProvider) : Int16;

10 IConvertible.ToInt32

12 [C#] int IConvertible.ToInt32(IFormatProvider provider);

13 [C++] int IConvertible::ToInt32(IFormatProvider* provider);

14 [VB] Function ToInt32(ByVal provider As IFormatProvider) As Integer

15 Implements IConvertible.ToInt32

16 [JScript] function IConvertible.ToInt32(provider : IFormatProvider) : int;

17 IConvertible.ToInt64

19 [C#] long IConvertible.ToInt64(IFormatProvider provider);

20 [C++] __int64 IConvertible::ToInt64(IFormatProvider* provider);

21 [VB] Function ToInt64(ByVal provider As IFormatProvider) As Long Implements

22 IConvertible.ToInt64

23 [JScript] function IConvertible.ToInt64(provider : IFormatProvider) : long;

24 IConvertible.ToSByte

```

1
2 [C#] sbyte IConvertible.ToSByte(IFormatProvider provider);
3 [C++] char IConvertible::ToSByte(IFormatProvider* provider);
4 [VB] Function ToSByte(ByVal provider As IFormatProvider) As SByte
5 Implements IConvertible.ToSByte
6 [JScript] function IConvertible.ToSByte(provider : IFormatProvider) : SByte;
7     IConvertible.ToSingle
8
9 [C#] float IConvertible.ToSingle(IFormatProvider provider);
10 [C++] float IConvertible::ToSingle(IFormatProvider* provider);
11 [VB] Function ToSingle(ByVal provider As IFormatProvider) As Single
12 Implements IConvertible.ToSingle
13 [JScript] function IConvertible.ToSingle(provider : IFormatProvider) : float;
14     IConvertible.ToType
15
16 [C#] object IConvertible.ToType(Type type, IFormatProvider provider);
17 [C++] Object* IConvertible::ToType(Type* type, IFormatProvider* provider);
18 [VB] Function ToType(ByVal type As Type, ByVal provider As IFormatProvider)
19 As Object Implements IConvertible.ToType
20 [JScript] function IConvertible.ToType(type : Type, provider : IFormatProvider) :
21 Object;
22     IConvertible.ToUInt16
23
24 [C#] ushort IConvertible.ToUInt16(IFormatProvider provider);
25 [C++] unsigned short IConvertible::ToUInt16(IFormatProvider* provider);

```



```

1 [VB] Function ToUInt16(ByVal provider As IFormatProvider) As UInt16
2 Implements IConvertible.ToUInt16
3 [JScript] function IConvertible.ToUInt16(provider : IFormatProvider) : UInt16;
4     IConvertible.ToUInt32
5
6 [C#] uint IConvertible.ToUInt32(IFormatProvider provider);
7 [C++] unsigned int IConvertible::ToUInt32(IFormatProvider* provider);
8 [VB] Function ToUInt32(ByVal provider As IFormatProvider) As UInt32
9 Implements IConvertible.ToUInt32
10 [JScript] function IConvertible.ToUInt32(provider : IFormatProvider) : UInt32;
11     IConvertible.ToUInt64
12
13 [C#] ulong IConvertible.ToUInt64(IFormatProvider provider);
14 [C++] unsigned __int64 IConvertible::ToUInt64(IFormatProvider* provider);
15 [VB] Function ToUInt64(ByVal provider As IFormatProvider) As UInt64
16 Implements IConvertible.ToUInt64
17 [JScript] function IConvertible.ToUInt64(provider : IFormatProvider) : UInt64;
18     ToFileTime
19
20 [C#] public long ToFileTime();
21 [C++] public: __int64 ToFileTime();
22 [VB] Public Function ToFileTime() As Long
23 [JScript] public function ToFileTime() : long;
24
25

```

Description

Converts the value of this instance to the format of the local system file time.

Return Value: The value of this **DateTime** in the format of the local system file time.

A system file time is a 64-bit unsigned value representing the date and time as the number of 100-nanosecond intervals that have elapsed since 1/1/1601 12:00 AM.

ToLocalTime

[C#] public DateTime ToLocalTime();

[C++] public: DateTime ToLocalTime();

[VB] Public Function ToLocalTime() As DateTime

[JScript] public function ToLocalTime() : DateTime;

Description

Converts the current coordinated universal time (UTC) to local time.

Return Value: The **DateTime** equivalent to the current UTC time, adjusted to the local time zone and daylight saving time.

This method always uses the local time zone when making calculations.

ToLongDateString

[C#] public string ToLongDateString();

[C++] public: String* ToLongDateString();

[VB] Public Function ToLongDateString() As String

[JScript] public function ToLongDateString() : String;

1
2 *Description*

3 Converts the date denoted by this instance to its equivalent long date **String**
4 representation.

5 *Return Value:* A **String** containing the name of the day of the week, the name of
6 the month, the numeric day of the month, and the year equivalent to the date value
7 of this instance.

8 The value of this instance is formatted using the long date format character,
9 'D'.

10 ToLongTimeString

11
12 [C#] public string ToLongTimeString();

13 [C++] public: String* ToLongTimeString();

14 [VB] Public Function ToLongTimeString() As String

15 [JScript] public function ToLongTimeString() : String;

16
17 *Description*

18 Converts the time denoted by this instance to its equivalent long time
19 **String** representation.

20 *Return Value:* A **String** containing the name of the day of the week, the name of
21 the month, the numeric day of the hours, minutes, and seconds equivalent to the
22 time value of this instance.

23 The value of this instance is formatted using the long time format character,
24 'T'.

25 ToOADate

```

1
2 [C#] public double ToOADate();
3 [C++] public: double ToOADate();
4 [VB] Public Function ToOADate() As Double
5 [JScript] public function ToOADate() : double;
6

```

7 *Description*

8 Converts the value of this instance to the equivalent OLE Automation date.

9 *Return Value:* A double-precision floating point number that contains an OLE
10 Automation date equivalent to the value of this instance.

11 An OLE Automation date is implemented as a floating-point number whose
12 value is the number of days from midnight, 30 December 1899. For example,
13 midnight, 31 December 1899 is represented by 1.0; 6 AM, 1 January 1900 is
14 represented by 2.25; midnight, 29 December 1899 represented by -1.0; and 6 AM,
15 29 December 1899 represented by -1.25.

16 *ToShortDateString*

```

17
18 [C#] public string ToShortDateString();
19 [C++] public: String* ToShortDateString();
20 [VB] Public Function ToShortDateString() As String
21 [JScript] public function ToShortDateString() : String;
22

```

23 *Description*

24 Converts the date denoted by this instance to its equivalent short date
25 **String** representation.

1 *Return Value:* A **String** containing the numeric month, the numeric day of the
2 month, and the year equivalent to the date value of this instance.

3 The value of this instance is formatted using the short date format character,
4 'd'.

5 ToShortTimeString

6
7 [C#] public string ToShortTimeString();

8 [C++] public: String* ToShortTimeString();

9 [VB] Public Function ToShortTimeString() As String

10 [JScript] public function ToShortTimeString() : String;

11
12 *Description*

13 Converts the time denoted by this instance to its equivalent short time
14 **String** representation.

15 *Return Value:* A **String** containing the name of the day of the week, the name of
16 the month, the numeric day of the hours, minutes, and seconds equivalent to the
17 time value of this instance.

18 The value of this instance is formatted using the short time format
19 character, 't'.

20 ToString

21
22 [C#] public override string ToString();

23 [C++] public: String* ToString();

24 [VB] Overrides Public Function ToString() As String

25 [JScript] public override function ToString() : String; Converts the value of this

instance to its equivalent **String** representation.

Description

Converts the value of this instance to its equivalent **String** .

Return Value: A **String** representation of value of this instance.

The value of this instance is formatted using the general format specifier, 'G', as described in the topic. The return value is of the following form:
MM/dd/yyyy HH:mm:ss The items in the return value are as follows.

ToString

[C#] public string ToString(IFormatProvider provider);

[C++] public: __sealed String* ToString(IFormatProvider* provider);

[VB] NotOverridable Public Function ToString(ByVal provider As IFormatProvider) As String

[JScript] public function ToString(provider : IFormatProvider) : String;

Description

Converts the value of this instance to its equivalent **String** representation using the specified culture-specific format information.

Return Value: A **String** representation of value of this instance as specified by *provider* .

This instance is formatted with the general format specifier, 'G'. An **System.IFormatProvider** interface implementation that supplies culture-specific formatting information.

ToString

```

1
2 [C#] public string ToString(string format);
3 [C++] public: String* ToString(String* format);
4 [VB] Public Function ToString(ByVal format As String) As String
5 [JScript] public function ToString(format : String) : String;
6

```

Description

Converts the value of this instance to its equivalent **String** representation using the specified format.

Return Value: A **String** representation of value of this instance as specified by *format* .

The format parmeter should contain either a format specifier character or a custom format pattern. For more information, see the summary page for **System.Globalization.DateTimeFormatInfo** . A format string.

ToString

```

17 [C#] public string ToString(string format, IFormatProvider provider);
18 [C++] public: __sealed String* ToString(String* format, IFormatProvider*
19 provider);
20 [VB] NotOverridable Public Function ToString(ByVal format As String, ByVal
21 provider As IFormatProvider) As String
22 [JScript] public function ToString(format : String, provider : IFormatProvider) :
23 String;
24

```

Description

Converts the value of this instance to its equivalent **String** representation using the specified format and culture-specific format information.

Return Value: A **String** representation of value of this instance as specified by *format* and *provider* .

The format parameter should contain either a format specifier character or a custom format pattern. For more information, see the summary page for **System.Globalization.DateTimeFormatInfo** . A format string. An **System.IFormatProvider** interface implementation that supplies culture-specific formatting information.

ToUniversalTime

[C#] public DateTime ToUniversalTime();

[C++] public: DateTime ToUniversalTime();

[VB] Public Function ToUniversalTime() As DateTime

[JScript] public function ToUniversalTime() : DateTime;

Description

Converts the current local time to coordinated universal time (UTC).

Return Value: The UTC **DateTime** equivalent to the current local time.

The UTC time is equal to the local time minus the UTC offset. For more information about the UTC offset, see

System.TimeZone.GetUtcOffset(System.DateTime) .

DayOfWeek enumeration (System)

ToUniversalTime

1
2
3 *Description*

4 Specifies the day of the week.

5 The **DayOfWeek** enumeration represents the day of the week in calendars
6 that have seven days per week. This enumeration ranges from zero, indicating
7 Sunday, to six, indicating Saturday.

8 ToUniversalTime

9
10 [C#] public const DayOfWeek Friday;

11 [C++] public: const DayOfWeek Friday;

12 [VB] Public Const Friday As DayOfWeek

13 [JScript] public var Friday : DayOfWeek;

14
15 *Description*

16 Indicates Friday.

17 ToUniversalTime

18
19 [C#] public const DayOfWeek Monday;

20 [C++] public: const DayOfWeek Monday;

21 [VB] Public Const Monday As DayOfWeek

22 [JScript] public var Monday : DayOfWeek;

23
24 *Description*

25 Indicates Monday.

ToUniversalTime

[C#] public const DayOfWeek Saturday;
[C++] public: const DayOfWeek Saturday;
[VB] Public Const Saturday As DayOfWeek
[JScript] public var Saturday : DayOfWeek;

Description

Indicates Saturday.

ToUniversalTime

[C#] public const DayOfWeek Sunday;
[C++] public: const DayOfWeek Sunday;
[VB] Public Const Sunday As DayOfWeek
[JScript] public var Sunday : DayOfWeek;

Description

Indicates Sunday.

ToUniversalTime

[C#] public const DayOfWeek Thursday;
[C++] public: const DayOfWeek Thursday;
[VB] Public Const Thursday As DayOfWeek
[JScript] public var Thursday : DayOfWeek;

1
2 *Description*

3 Indicates Thursday.

4 ToUniversalTime

5
6 [C#] public const DayOfWeek Tuesday;

7 [C++] public: const DayOfWeek Tuesday;

8 [VB] Public Const Tuesday As DayOfWeek

9 [JScript] public var Tuesday : DayOfWeek;

10
11 *Description*

12 Indicates Tuesday.

13 ToUniversalTime

14
15 [C#] public const DayOfWeek Wednesday;

16 [C++] public: const DayOfWeek Wednesday;

17 [VB] Public Const Wednesday As DayOfWeek

18 [JScript] public var Wednesday : DayOfWeek;

19
20 *Description*

21 Indicates Wednesday.

22 DBNull class (System)

23 ToString

Description

Represents a null value.

This class is used to indicate the absence of a known value, typically in a database application.

ToString

[C#] public static readonly DBNull Value;

[C++] public: static DBNull* Value;

[VB] Public Shared ReadOnly Value As DBNull

[JScript] public static var Value : DBNull;

Description

Represents the sole instance of the **System.DBNull** class.

System.DBNull is a singleton class, which means only this instance of this class can exist.

GetObjectData

[C#] public void GetObjectData(SerializationInfo info, StreamingContext context);

[C++] public: __sealed void GetObjectData(SerializationInfo* info, StreamingContext context);

[VB] NotOverridable Public Sub GetObjectData(ByVal info As SerializationInfo, ByVal context As StreamingContext)

1 [JScript] public function GetObjectData(info : SerializationInfo, context :
2 StreamingContext);

3
4 *Description*

5 Implements the **System.Runtime.Serialization.ISerializable** interface and
6 returns the data needed to serialize the **System.DBNull** object. A

7 **System.Runtime.Serialization.SerializationInfo** object containing information
8 required to serialize the **System.DBNull** object. A

9 **System.Runtime.Serialization.StreamingContext** object containing the source
10 and destination of the serialized stream associated with the **System.DBNull** object.

11 GetTypeCode

12
13 [C#] public TypeCode GetTypeCode();

14 [C++] public: __sealed TypeCode GetTypeCode();

15 [VB] NotOverridable Public Function GetTypeCode() As TypeCode

16 [JScript] public function GetTypeCode() : TypeCode;

17
18 *Description*

19 Gets the **System.TypeCode** value for **System.DBNull** .

20 *Return Value:* The **System.TypeCode** value for **System.DBNull** , which is
21 **System.TypeCode.DBNull** .

22 IConvertible.ToBoolean

23
24 [C#] bool IConvertible.ToBoolean(IFormatProvider provider);

25 [C++] bool IConvertible::ToBoolean(IFormatProvider* provider);

```

1  [VB] Function ToBoolean(ByVal provider As IFormatProvider) As Boolean
2  Implements IConvertible.ToBoolean
3  [JScript] function IConvertible.ToBoolean(provider : IFormatProvider) : Boolean;
4      IConvertible.ToByte
5
6  [C#] byte IConvertible.ToByte(IFormatProvider provider);
7  [C++] unsigned char IConvertible::ToByte(IFormatProvider* provider);
8  [VB] Function ToByte(ByVal provider As IFormatProvider) As Byte Implements
9  IConvertible.ToByte
10 [JScript] function IConvertible.ToByte(provider : IFormatProvider) : Byte;
11     IConvertible.ToChar
12
13 [C#] char IConvertible.ToChar(IFormatProvider provider);
14 [C++] __wchar_t IConvertible::ToChar(IFormatProvider* provider);
15 [VB] Function ToChar(ByVal provider As IFormatProvider) As Char Implements
16 IConvertible.ToChar
17 [JScript] function IConvertible.ToChar(provider : IFormatProvider) : Char;
18     IConvertible.ToDateTime
19
20 [C#] DateTime IConvertible.ToDateTime(IFormatProvider provider);
21 [C++] DateTime IConvertible::ToDateTime(IFormatProvider* provider);
22 [VB] Function ToDateTime(ByVal provider As IFormatProvider) As DateTime
23 Implements IConvertible.ToDateTime
24 [JScript] function IConvertible.ToDateTime(provider : IFormatProvider) :
25 DateTime;
```

1 IConvertible.ToDecimal

3 [C#] decimal IConvertible.ToDecimal(IFormatProvider provider);

4 [C++] Decimal IConvertible::ToDecimal(IFormatProvider* provider);

5 [VB] Function ToDecimal(ByVal provider As IFormatProvider) As Decimal

6 Implements IConvertible.ToDecimal

7 [JScript] function IConvertible.ToDecimal(provider : IFormatProvider) : Decimal;

8 IConvertible.ToDouble

10 [C#] double IConvertible.ToDouble(IFormatProvider provider);

11 [C++] double IConvertible::ToDouble(IFormatProvider* provider);

12 [VB] Function ToDouble(ByVal provider As IFormatProvider) As Double

13 Implements IConvertible.ToDouble

14 [JScript] function IConvertible.ToDouble(provider : IFormatProvider) : double;

15 IConvertible.ToInt16

17 [C#] short IConvertible.ToInt16(IFormatProvider provider);

18 [C++] short IConvertible::ToInt16(IFormatProvider* provider);

19 [VB] Function ToInt16(ByVal provider As IFormatProvider) As Short

20 Implements IConvertible.ToInt16

21 [JScript] function IConvertible.ToInt16(provider : IFormatProvider) : Int16;

22 IConvertible.ToInt32

24 [C#] int IConvertible.ToInt32(IFormatProvider provider);

25 [C++] int IConvertible::ToInt32(IFormatProvider* provider);

```

1 [VB] Function ToInt32(ByVal provider As IFormatProvider) As Integer
2 Implements IConvertible.ToInt32
3 [JScript] function IConvertible.ToInt32(provider : IFormatProvider) : int;
4     IConvertible.ToInt64
5
6 [C#] long IConvertible.ToInt64(IFormatProvider provider);
7 [C++] __int64 IConvertible::ToInt64(IFormatProvider* provider);
8 [VB] Function ToInt64(ByVal provider As IFormatProvider) As Long Implements
9 IConvertible.ToInt64
10 [JScript] function IConvertible.ToInt64(provider : IFormatProvider) : long;
11     IConvertible.ToSByte
12
13 [C#] sbyte IConvertible.ToSByte(IFormatProvider provider);
14 [C++] char IConvertible::ToSByte(IFormatProvider* provider);
15 [VB] Function ToSByte(ByVal provider As IFormatProvider) As SByte
16 Implements IConvertible.ToSByte
17 [JScript] function IConvertible.ToSByte(provider : IFormatProvider) : SByte;
18     IConvertible.ToSingle
19
20 [C#] float IConvertible.ToSingle(IFormatProvider provider);
21 [C++] float IConvertible::ToSingle(IFormatProvider* provider);
22 [VB] Function ToSingle(ByVal provider As IFormatProvider) As Single
23 Implements IConvertible.ToSingle
24 [JScript] function IConvertible.ToSingle(provider : IFormatProvider) : float;
25     IConvertible.ToType

```



```

1
2 [C#] object IConvertible.ToType(Type type, IFormatProvider provider);
3 [C++] Object* IConvertible::ToType(Type* type, IFormatProvider* provider);
4 [VB] Function ToType(ByVal type As Type, ByVal provider As IFormatProvider)
5 As Object Implements IConvertible.ToType
6 [JScript] function IConvertible.ToType(type : Type, provider : IFormatProvider) :
7 Object;
8     IConvertible.ToUInt16
9
10 [C#] ushort IConvertible.ToUInt16(IFormatProvider provider);
11 [C++] unsigned short IConvertible::ToUInt16(IFormatProvider* provider);
12 [VB] Function ToUInt16(ByVal provider As IFormatProvider) As UInt16
13 Implements IConvertible.ToUInt16
14 [JScript] function IConvertible.ToUInt16(provider : IFormatProvider) : UInt16;
15     IConvertible.ToUInt32
16
17 [C#] uint IConvertible.ToUInt32(IFormatProvider provider);
18 [C++] unsigned int IConvertible::ToUInt32(IFormatProvider* provider);
19 [VB] Function ToUInt32(ByVal provider As IFormatProvider) As UInt32
20 Implements IConvertible.ToUInt32
21 [JScript] function IConvertible.ToUInt32(provider : IFormatProvider) : UInt32;
22     IConvertible.ToUInt64
23
24 [C#] ulong IConvertible.ToUInt64(IFormatProvider provider);
25 [C++] unsigned __int64 IConvertible::ToUInt64(IFormatProvider* provider);

```

1 [VB] Function ToUInt64(ByVal provider As IFormatProvider) As UInt64
 2 Implements IConvertible.ToUInt64
 3 [JScript] function IConvertible.ToUInt64(provider : IFormatProvider) : UInt64;
 4 ToString
 5
 6 [C#] public override string ToString();
 7 [C++] public: String* ToString();
 8 [VB] Overrides Public Function ToString() As String
 9 [JScript] public override function ToString() : String; Returns an empty string.

11 *Description*

12 Returns an empty string (**System.String.Empty**).

13 *Return Value:* An empty string (**System.String.Empty**).

14 ToString

15
 16 [C#] public string ToString(IFormatProvider provider);
 17 [C++] public: __sealed String* ToString(IFormatProvider* provider);
 18 [VB] NotOverridable Public Function ToString(ByVal provider As
 19 IFormatProvider) As String
 20 [JScript] public function ToString(provider : IFormatProvider) : String;

22 *Description*

23 Returns an empty string using the specified **System.IFormatProvider** .

24 *Return Value:* An empty string (**System.String.Empty**). The

25 **System.IFormatProvider** to be used to format the string.

Decimal structure (System)

ToString

Description

Represents a decimal number.

The **Decimal** value type represents decimal numbers ranging from positive 79,228,162,514,264,337,593,543,950,335 to negative 79,228,162,514,264,337,593,543,950,335. The **Decimal** value type is appropriate for financial calculations requiring large numbers of significant integral and fractional digits and no round-off errors.

ToString

[C#] public static readonly decimal MaxValue;

[C++] public: static Decimal MaxValue;

[VB] Public Shared ReadOnly MaxValue As Decimal

[JScript] public static var MaxValue : Decimal;

Description

A constant representing the largest possible value of **Decimal** .

The value of this constant is positive

79,228,162,514,264,337,593,543,950,335.

ToString

[C#] public static readonly decimal MinusOne;

[C++] public: static Decimal MinusOne;
 [VB] Public Shared ReadOnly MinusOne As Decimal
 [JScript] public static var MinusOne : Decimal;

Description

A constant representing the number, negative one.

ToString

[C#] public static readonly decimal MinValue;
 [C++] public: static Decimal MinValue;
 [VB] Public Shared ReadOnly MinValue As Decimal
 [JScript] public static var MinValue : Decimal;

Description

A constant representing the smallest possible value of **Decimal** .

The value of this constant is negative

79,228,162,514,264,337,593,543,950,335.

ToString

[C#] public static readonly decimal One;
 [C++] public: static Decimal One;
 [VB] Public Shared ReadOnly One As Decimal
 [JScript] public static var One : Decimal;

Description

A constant representing the number, one.

ToString

[C#] public static readonly decimal Zero;

[C++] public: static Decimal Zero;

[VB] Public Shared ReadOnly Zero As Decimal

[JScript] public static var Zero : Decimal;

Description

A constant representing the number, zero.

Decimal

Example Syntax:

ToString

[C#] public Decimal(double value);

[C++] public: Decimal(double value);

[VB] Public Sub New(ByVal value As Double)

[JScript] public function Decimal(value : double);

Description

Initializes a new instance of **Decimal** to the value of the specified double-precision floating point number. The value to represent as a **Decimal**.

Decimal

Example Syntax:

ToString

1
2 [C#] public Decimal(int value);

3 [C++] public: Decimal(int value);

4 [VB] Public Sub New(ByVal value As Integer)

5 [JScript] public function Decimal(value : int); Initializes a new instance of

6 **Decimal** .

7
8 *Description*

9 Initializes a new instance of **Decimal** to the value of the specified 32-bit
10 signed integer. The value to represent as a **Decimal**.

11 Decimal

12 *Example Syntax:*

13 ToString

14
15 [C#] public Decimal(int[] bits);

16 [C++] public: Decimal(int bits __gc[]);

17 [VB] Public Sub New(ByVal bits() As Integer)

18 [JScript] public function Decimal(bits : int[]);

19
20 *Description*

21 Initializes a new instance of **Decimal** to a decimal value represented in
22 binary and contained in a specified array.

23 The binary representation of a **Decimal** number consists of a 1-bit sign, a
24 96-bit integer number, and a scaling factor used to divide the integer number and
25 specify what portion of it is a decimal fraction. The scaling factor is implicitly the

number 10, raised to an exponent ranging from 0 to 28. An array of 32-bit signed integers containing a representation of a decimal value.

Decimal

Example Syntax:

ToString

[C#] public Decimal(long value);

[C++] public: Decimal(__int64 value);

[VB] Public Sub New(ByVal value As Long)

[JScript] public function Decimal(value : long);

Description

Initializes a new instance of **Decimal** to the value of the specified 64-bit signed integer. The value to represent as a **Decimal**.

Decimal

Example Syntax:

ToString

[C#] public Decimal(float value);

[C++] public: Decimal(float value);

[VB] Public Sub New(ByVal value As Single)

[JScript] public function Decimal(value : float);

Description

1 Initializes a new instance of **Decimal** to the value of the specified single-
2 precision floating point number. The value to represent as a **Decimal**.

3 Decimal

4 *Example Syntax:*

5 ToString

6
7 [C#] public Decimal(uint value);

8 [C++] public: Decimal(unsigned int value);

9 [VB] Public Sub New(ByVal value As UInt32)

10 [JScript] public function Decimal(value : UInt32);

11
12 *Description*

13 Initializes a new instance of **Decimal** to the value of the specified 32-bit
14 unsigned integer. The value to represent as a **Decimal**.

15 Decimal

16 *Example Syntax:*

17 ToString

18
19 [C#] public Decimal(ulong value);

20 [C++] public: Decimal(unsigned __int64 value);

21 [VB] Public Sub New(ByVal value As UInt64)

22 [JScript] public function Decimal(value : UInt64);

23
24 *Description*

1 Initializes a new instance of **Decimal** to the value of the specified 64-bit
2 unsigned integer. The value to represent as a **Decimal**.

3 **Decimal**

4 *Example Syntax:*

5 **ToString**

6
7 [C#] public Decimal(int lo, int mid, int hi, bool isNegative, byte scale);

8 [C++] public: Decimal(int lo, int mid, int hi, bool isNegative, unsigned char
9 scale);

10 [VB] Public Sub New(ByVal lo As Integer, ByVal mid As Integer, ByVal hi As
11 Integer, ByVal isNegative As Boolean, ByVal scale As Byte)

12 [JScript] public function Decimal(lo : int, mid : int, hi : int, isNegative : Boolean,
13 scale : Byte);

14
15 *Description*

16 Initializes a new instance of **Decimal** from parameters specifying the
17 instance's constituent parts.

18 The binary representation of a **Decimal** number consists of a 1-bit sign, a
19 96-bit integer number, and a scaling factor used to divide the integer number and
20 specify what portion of it is a decimal fraction. The scaling factor is implicitly the
21 number 10, raised to an exponent ranging from 0 to 28. The low 32 bits of a 96-bit
22 integer. The middle 32 bits of a 96-bit integer. The high 32 bits of a 96-bit integer.
23 The sign; 1 is negative, 0 is positive. A power of 10 ranging from 0 to 28.

24 **Add**

```

1
2 [C#] public static decimal Add(decimal d1, decimal d2);
3 [C++] public: static Decimal Add(Decimal d1, Decimal d2);
4 [VB] Public Shared Function Add(ByVal d1 As Decimal, ByVal d2 As Decimal)
5 As Decimal
6 [JScript] public static function Add(d1 : Decimal, d2 : Decimal) : Decimal;
7

```

Description

Adds two specified **Decimal** values.

Return Value: A **Decimal** value that is the sum of *d1* and *d2* . A **Decimal**. A **Decimal**.

Compare

```

13
14 [C#] public static int Compare(decimal d1, decimal d2);
15 [C++] public: static int Compare(Decimal d1, Decimal d2);
16 [VB] Public Shared Function Compare(ByVal d1 As Decimal, ByVal d2 As
17 Decimal) As Integer
18 [JScript] public static function Compare(d1 : Decimal, d2 : Decimal) : int;
19

```

Description

Compares two specified **Decimal** values.

Return Value: A signed number indicating the relative values of *d1* and *d2* . A **Decimal**. A **Decimal**.

CompareTo

1
2 [C#] public int CompareTo(object value);

3 [C++] public: __sealed int CompareTo(Object* value);

4 [VB] NotOverridable Public Function CompareTo(ByVal value As Object) As

5 Integer

6 [JScript] public function CompareTo(value : Object) : int;

7
8 *Description*

9 Compares this instance to a specified **Object** .

10 *Return Value:* A signed number indicating the relative values of this instance and
11 *value* .

12 Any instance of **Decimal** , regardless of its value, is considered greater than
13 **null** . An **System.Object** or **null**.

14 Divide

15
16 [C#] public static decimal Divide(decimal d1, decimal d2);

17 [C++] public: static Decimal Divide(Decimal d1, Decimal d2);

18 [VB] Public Shared Function Divide(ByVal d1 As Decimal, ByVal d2 As

19 Decimal) As Decimal

20 [JScript] public static function Divide(d1 : Decimal, d2 : Decimal) : Decimal;

21
22 *Description*

23 Divides two specified **Decimal** values.

24 *Return Value:* The **Decimal** that is the result of dividing *d1* by *d2* . A **Decimal**
25 (the dividend). A **Decimal** (the divisor).

Equals

[C#] public override bool Equals(object value);

[C++] public: bool Equals(Object* value);

[VB] Overrides Public Function Equals(ByVal value As Object) As Boolean

[JScript] public override function Equals(value : Object) : Boolean; Returns a

value indicating whether two instances of **Decimal** are equal.

Description

Returns a value indicating whether this instance and a specified **Object** are equal.

Return Value: **true** if *value* is a **Decimal** and equal to this instance; otherwise, **false**. An **System.Object**.

Equals

[C#] public static new bool Equals(decimal d1, decimal d2);

[C++] public: static bool Equals(Decimal d1, Decimal d2);

[VB] Shadows Public Shared Function Equals(ByVal d1 As Decimal, ByVal d2

As Decimal) As Boolean

[JScript] public static hide function Equals(d1 : Decimal, d2 : Decimal) : Boolean;

Description

Returns a value indicating whether two specified instances of **Decimal** are equal.

1 *Return Value:* **true** if *d1* and *d2* are equal; otherwise, *false* . A **Decimal**. A

2 **Decimal.**

3 Floor

4
5 [C#] public static decimal Floor(decimal d);

6 [C++] public: static Decimal Floor(Decimal d);

7 [VB] Public Shared Function Floor(ByVal d As Decimal) As Decimal

8 [JScript] public static function Floor(d : Decimal) : Decimal;

9
10 *Description*

11 Rounds a specified **Decimal** number to the next lower whole number.

12 *Return Value:* If *d* has a fractional part, the next whole **Decimal** number towards
13 negative infinity that is less than *d*. -or- If *d* doesn't have a fractional part, *d* is
14 returned unchanged. A **Decimal**.

15 FromOACurrency

16
17 [C#] public static decimal FromOACurrency(long cy);

18 [C++] public: static Decimal FromOACurrency(__int64 cy);

19 [VB] Public Shared Function FromOACurrency(ByVal cy As Long) As Decimal

20 [JScript] public static function FromOACurrency(cy : long) : Decimal;

21
22 *Description*

23 Converts the specified 64-bit signed integer, which contains an OLE
24 Automation Currency value, to the equivalent **Decimal** value.

1 *Return Value:* A **Decimal** that contains the equivalent of *cy* . An OLE Automation
2 Currency value.

3 GetBits

4
5 [C#] public static int[] GetBits(decimal d);

6 [C++] public: static int GetBits(Decimal d) __gc[];

7 [VB] Public Shared Function GetBits(ByVal d As Decimal) As Integer()

8 [JScript] public static function GetBits(d : Decimal) : int[];

9 10 *Description*

11 Converts the value of a specified instance of **Decimal** to its equivalent
12 binary representation, and returns that representation in an array of 32-bit signed
13 integers.

14 *Return Value:* A 32-bit integer array with four elements that contain the binary
15 representation of *d* .

16 The binary representation of a **Decimal** number consists of a 1-bit sign, a
17 96-bit integer number, and a scaling factor used to divide the integer number and
18 specify what portion of it is a decimal fraction. The scaling factor is implicitly the
19 number 10, raised to an exponent ranging from 0 to 28. A **Decimal** value.

20 GetHashCode

21
22 [C#] public override int GetHashCode();

23 [C++] public: int GetHashCode();

24 [VB] Overrides Public Function GetHashCode() As Integer

25 [JScript] public override function GetHashCode() : int;

Description

Returns the hash code for this instance.

Return Value: A 32-bit signed integer hash code.

GetTypeCode

[C#] public TypeCode GetTypeCode();

[C++] public: __sealed TypeCode GetTypeCode();

[VB] NotOverridable Public Function GetTypeCode() As TypeCode

[JScript] public function GetTypeCode() : TypeCode;

Description

Returns the **TypeCode** for value type **Decimal** .

Return Value: The enumerated constant, **System.TypeCode.Decimal** .

Multiply

[C#] public static decimal Multiply(decimal d1, decimal d2);

[C++] public: static Decimal Multiply(Decimal d1, Decimal d2);

[VB] Public Shared Function Multiply(ByVal d1 As Decimal, ByVal d2 As
Decimal) As Decimal

[JScript] public static function Multiply(d1 : Decimal, d2 : Decimal) : Decimal;

Description

Multiplies two specified **Decimal** values.

Return Value: A **Decimal** that is the result of multiplying *d1* and *d2* . A **Decimal** (the multiplicand). A **Decimal** (the multiplier).

Negate

[C#] public static decimal Negate(decimal d);

[C++] public: static Decimal Negate(Decimal d);

[VB] Public Shared Function Negate(ByVal d As Decimal) As Decimal

[JScript] public static function Negate(d : Decimal) : Decimal;

Description

Negates the value of a specified **Decimal** .

Return Value: A **Decimal** with the value of *d* , but the opposite sign.

The Negate method returns the negative of the specified Decimal value. For example: If *d* is non-zero, the result is *-d* . A **Decimal**.

op_Addition

[C#] public static decimal operator +(decimal d1, decimal d2);

[C++] public: static Decimal op_Addition(Decimal d1, Decimal d2);

[VB] returnValue = Decimal.op_Addition(d1, d2)

[JScript] returnValue = d1 + d2;

Description

Adds two specified **Decimal** values.

Return Value: The **Decimal** result of adding *d1* and *d2* . A **Decimal**. A **Decimal**.

op_Decrement

[C#] public static decimal operator --(decimal d);
[C++] public: static Decimal op_Decrement(Decimal d);
[VB] returnValue = Decimal.op_Decrement(d)
[JScript] returnValue = d--;

Description

Decrements the **Decimal** operand by 1.

Return Value: The value of *d* decremented by 1. The **Decimal** operand.

op_Division

[C#] public static decimal operator /(decimal d1, decimal d2);
[C++] public: static Decimal op_Division(Decimal d1, Decimal d2);
[VB] returnValue = Decimal.op_Division(d1, d2)
[JScript] returnValue = d1 / d2;

Description

Divides two specified **Decimal** values.

Return Value: The **Decimal** result of *d1* by *d2*. A **Decimal** (the dividend). A **Decimal** (the divisor).

op_Equality

[C#] public static bool operator ==(decimal d1, decimal d2);
[C++] public: static bool op_Equality(Decimal d1, Decimal d2);

1 [VB] returnValue = Decimal.op_Equality(d1, d2)

2 [JScript] returnValue = d1 == d2;

3
4 *Description*

5 Returns a value indicating whether two instances of **Decimal** are equal.

6 *Return Value:* **true** if *d1* and *d2* are equal; otherwise, **false** . A **Decimal**. A

7 **Decimal.**

8 op_Explicit

9
10 [C#] public static explicit operator ushort(decimal value);

11 [C++] public: static unsigned short op_Explicit();

12 [VB] returnValue = Decimal.op_Explicit(value)

13 [JScript] returnValue = UInt16(value);

14
15 *Description*

16 Converts a **Decimal** to a 16-bit unsigned integer.

17 *Return Value:* A 16-bit unsigned integer that represents the converted **Decimal** . A

18 **Decimal** to convert.

19 op_Explicit

20
21 [C#] public static explicit operator int(decimal value);

22 [C++] public: static int op_Explicit();

23 [VB] returnValue = Decimal.op_Explicit(value)

24 [JScript] returnValue = Int32(value);

1
2 *Description*

3 Converts a **Decimal** to a 32-bit signed integer.

4 *Return Value:* A 32-bit signed integer that represents the converted **Decimal** . A

5 **Decimal** to convert.

6 op_Explicit

7
8 [C#] public static explicit operator byte(decimal value);

9 [C++] public: static unsigned char op_Explicit();

10 [VB] returnValue = Decimal.op_Explicit(value)

11 [JScript] returnValue = Byte(value);

12
13 *Description*

14 Converts a **Decimal** to an 8-bit unsigned integer.

15 *Return Value:* An 8-bit unsigned integer that represents the converted **Decimal** . A

16 **Decimal** to convert.

17 op_Explicit

18
19 [C#] public static explicit operator sbyte(decimal value);

20 [C++] public: static char op_Explicit();

21 [VB] returnValue = Decimal.op_Explicit(value)

22 [JScript] returnValue = SByte(value);

23
24 *Description*

Converts a **Decimal** to an 8-bit signed integer.

Return Value: An 8-bit signed integer that represents the converted **Decimal** . A

Decimal to convert.

op_Explicit

[C#] public static explicit operator char(decimal value);

[C++] public: static __wchar_t op_Explicit();

[VB] returnValue = Decimal.op_Explicit(value)

[JScript] returnValue = Char(value);

Description

Converts a **Decimal** to a Unicode character.

Return Value: A Unicode character that represents the converted **Decimal** . A

Decimal to convert.

op_Explicit

[C#] public static explicit operator short(decimal value);

[C++] public: static short op_Explicit();

[VB] returnValue = Decimal.op_Explicit(value)

[JScript] returnValue = Int16(value);

Description

Converts a **Decimal** to a 16-bit signed integer.

Return Value: A 16-bit signed integer that represents the converted **Decimal** . A

Decimal to convert.

```

1      op_Explicit
2
3  [C#] public static explicit operator float(decimal value);
4  [C++] public: static float op_Explicit();
5  [VB] returnValue = Decimal.op_Explicit(value)
6  [JScript] returnValue = Single(value);
7

```

8 *Description*

9 Converts a **Decimal** to a single-precision floating point number.

10 *Return Value:* A single-precision floating point number that represents the
 11 converted **Decimal** . A **Decimal** to convert.

```

12      op_Explicit
13

```

```

14  [C#] public static explicit operator double(decimal value);
15  [C++] public: static double op_Explicit();
16  [VB] returnValue = Decimal.op_Explicit(value)
17  [JScript] returnValue = Double(value);
18

```

19 *Description*

20 Converts a **Decimal** to a double-precision floating point number.

21 *Return Value:* A double-precision floating point number that represents the
 22 converted **Decimal** . A **Decimal** to convert.

```

23      op_Explicit
24

```

```

25  [C#] public static explicit operator ulong(decimal value);

```

1 [C++] public: static unsigned __int64 op_Explicit();

2 [VB] returnValue = Decimal.op_Explicit(value)

3 [JScript] returnValue = UInt64(value);

4
5 *Description*

6 Converts a **Decimal** to a 64-bit unsigned integer.

7 *Return Value:* A 64-bit unsigned integer that represents the converted **Decimal** . A
8 **Decimal** to convert.

9 op_Explicit

10
11 [C#] public static explicit operator uint(decimal value);

12 [C++] public: static unsigned int op_Explicit();

13 [VB] returnValue = Decimal.op_Explicit(value)

14 [JScript] returnValue = UInt32(value);

15
16 *Description*

17 Converts a **Decimal** to a 32-bit unsigned integer.

18 *Return Value:* A 32-bit unsigned integer that represents the converted **Decimal** . A
19 **Decimal** to convert.

20 op_Explicit

21
22 [C#] public static explicit operator long(decimal value);

23 [C++] public: static __int64 op_Explicit();

24 [VB] returnValue = Decimal.op_Explicit(value)

25 [JScript] returnValue = Int64(value);

1
2 *Description*

3 Converts a **Decimal** to a 64-bit signed integer.

4 *Return Value:* A 64-bit signed integer that represents the converted **Decimal** . A
5 **Decimal** to convert.

6 op_Explicit

7
8 [C#] public static explicit operator decimal(double value);

9 [C++] public: static Decimal op_Explicit(double value);

10 [VB] returnValue = Decimal.op_Explicit(value)

11 [JScript] returnValue = Decimal(value);

12
13 *Description*

14 Converts a double-precision floating point number to a **Decimal** .

15 *Return Value:* A **Decimal** that represents the converted double-precision floating
16 point number. A double-precision floating point number.

17 op_Explicit

18
19 [C#] public static explicit operator decimal(float value);

20 [C++] public: static Decimal op_Explicit(float value);

21 [VB] returnValue = Decimal.op_Explicit(value)

22 [JScript] returnValue = Decimal(value);

23
24 *Description*

Converts a single-precision floating point number to a **Decimal** .

Return Value: A **Decimal** that represents the converted single-precision floating point number. A single-precision floating point number.

op_GreaterThan

[C#] public static bool operator >(decimal d1, decimal d2);

[C++] public: static bool op_GreaterThan(Decimal d1, Decimal d2);

[VB] returnValue = Decimal.op_GreaterThan(d1, d2)

[JScript] returnValue = d1 > d2;

Description

Returns a value indicating whether a specified **Decimal** is greater than another specified **Decimal** .

Return Value: **true** if *d1* is greater than *d2* ; otherwise, **false** . A **Decimal**. A **Decimal**.

op_GreaterThanOrEqual

[C#] public static bool operator >=(decimal d1, decimal d2);

[C++] public: static bool op_GreaterThanOrEqual(Decimal d1, Decimal d2);

[VB] returnValue = Decimal.op_GreaterThanOrEqual(d1, d2)

[JScript] returnValue = d1 >= d2;

Description

Returns a value indicating whether a specified **Decimal** is greater than or equal to another specified **Decimal** .

Return Value: **true** if *d1* is greater than or equal to *d2* ; otherwise, **false** . A

Decimal. A Decimal.

op_Explicit

[C#] public static implicit operator decimal(byte value);

[C++] public: static Decimal op_Explicit(unsigned char value);

[VB] returnValue = Decimal.op_Explicit(value)

[JScript] returnValue = value;

Description

Converts an 8-bit unsigned integer to a **Decimal** .

Return Value: A **Decimal** that represents the converted 8-bit unsigned integer. An 8-bit unsigned integer.

op_Explicit

[C#] public static implicit operator decimal(char value);

[C++] public: static Decimal op_Explicit(__wchar_t value);

[VB] returnValue = Decimal.op_Explicit(value)

[JScript] returnValue = value;

Description

Converts a Unicode character to a **Decimal** .

Return Value: A **Decimal** that represents the converted Unicode character. A Unicode character.

op_Explicit

```

1
2 [C#] public static implicit operator decimal(short value);
3 [C++] public: static Decimal op_Implicit(short value);
4 [VB] returnValue = Decimal.op_Implicit(value)
5 [JScript] returnValue = value;

```

6 *Description*

7 Converts a 16-bit signed integer to a **Decimal** .

8 *Return Value:* A **Decimal** that represents the converted 16-bit signed integer. A
9 16-bit signed integer.

10 op_Implicit

```

11
12
13 [C#] public static implicit operator decimal(int value);
14 [C++] public: static Decimal op_Implicit(int value);
15 [VB] returnValue = Decimal.op_Implicit(value)
16 [JScript] returnValue = value;

```

17 *Description*

18 Converts a 32-bit signed integer to a **Decimal** .

19 *Return Value:* A **Decimal** that represents the converted 32-bit signed integer. A
20 32-bit signed integer.

21 op_Implicit

```

22
23
24 [C#] public static implicit operator decimal(long value);
25 [C++] public: static Decimal op_Implicit(__int64 value);

```

1 [VB] returnValue = Decimal.op_Implicit(value)

2 [JScript] returnValue = value;

3
4 *Description*

5 Converts a 64-bit signed integer to a **Decimal** .

6 *Return Value:* A **Decimal** that represents the converted 64-bit signed integer. A

7 64-bit signed integer.

8 op_Implicit

9
10 [C#] public static implicit operator decimal(sbyte value);

11 [C++] public: static Decimal op_Implicit(char value);

12 [VB] returnValue = Decimal.op_Implicit(value)

13 [JScript] returnValue = value;

14
15 *Description*

16 Converts an 8-bit signed integer to a **Decimal** .

17 *Return Value:* A **Decimal** that represents the converted 8-bit signed integer. An 8-

18 bit signed integer.

19 op_Implicit

20
21 [C#] public static implicit operator decimal(ushort value);

22 [C++] public: static Decimal op_Implicit(unsigned short value);

23 [VB] returnValue = Decimal.op_Implicit(value)

24 [JScript] returnValue = value;

25

1
2 *Description*

3 Converts a 16-bit unsigned integer to a **Decimal** .

4 *Return Value:* A **Decimal** that represents the converted 16-bit unsigned integer. A
5 16-bit unsigned integer.

6 op_Implicit

7
8 [C#] public static implicit operator decimal(uint value);

9 [C++] public: static Decimal op_Implicit(unsigned int value);

10 [VB] returnValue = Decimal.op_Implicit(value)

11 [JScript] returnValue = value;

12
13 *Description*

14 Converts a 32-bit unsigned integer to a **Decimal** .

15 *Return Value:* A **Decimal** that represents the converted 32-bit unsigned integer. A
16 32-bit unsigned integer.

17 op_Implicit

18
19 [C#] public static implicit operator decimal(ulong value);

20 [C++] public: static Decimal op_Implicit(unsigned __int64 value);

21 [VB] returnValue = Decimal.op_Implicit(value)

22 [JScript] returnValue = value;

23
24 *Description*
25

Converts a 64-bit unsigned integer to a **Decimal** .

Return Value: A **Decimal** that represents the converted 64-bit unsigned integer. A 64-bit unsigned integer.

op_Increment

[C#] public static decimal operator ++(decimal d);

[C++] public: static Decimal op_Increment(Decimal d);

[VB] returnValue = Decimal.op_Increment(d)

[JScript] returnValue = d++;

Description

Increments the **Decimal** operand by 1.

Return Value: The value of *d* incremented by 1. The **Decimal** operand.

op_Inequality

[C#] public static bool operator !=(decimal d1, decimal d2);

[C++] public: static bool op_Inequality(Decimal d1, Decimal d2);

[VB] returnValue = Decimal.op_Inequality(d1, d2)

[JScript] returnValue = d1 != d2;

Description

Returns a value indicating whether two instances of **Decimal** are not equal.

Return Value: **true** if *d1* and *d2* are not equal; otherwise, **false** . A **Decimal**. A

Decimal.

op_LessThan

```

1
2 [C#] public static bool operator
3 [C++] public: static bool op_LessThan(Decimal d1, Decimal d2);
4 [VB] returnValue = Decimal.op_LessThan(d1, d2)
5 [JScript] returnValue = d1 < d2;

```

6 *Description*

7 Returns a value indicating whether a specified **Decimal** is less than another
8 specified **Decimal** .

9 *Return Value:* **true** if *d1* is less than *d2* ; otherwise, **false** . A **Decimal**. A **Decimal**.

10 op_LessThanOrEqual

```

11
12
13 [C#] public static bool operator <=(decimal d1, decimal d2);
14 [C++] public: static bool op_LessThanOrEqual(Decimal d1, Decimal d2);
15 [VB] returnValue = Decimal.op_LessThanOrEqual(d1, d2)
16 [JScript] returnValue = d1 <= d2;

```

17 *Description*

18 Returns a value indicating whether a specified **Decimal** is less than or equal
19 to another specified **Decimal** .

20 *Return Value:* **true** if *d1* is less than or equal to *d2* ; otherwise, **false** . A **Decimal**.

21 A **Decimal**.

22 op_Modulus

```

23
24
25 [C#] public static decimal operator %(decimal d1, decimal d2);

```

1 [C++] public: static Decimal op_Modulus(Decimal d1, Decimal d2);

2 [VB] returnValue = Decimal.op_Modulus(d1, d2)

3 [JScript] returnValue = d1 % d2;

4
5 *Description*

6 Returns the remainder resulting from dividing two specified **Decimal**
7 values.

8 *Return Value:* The **Decimal** remainder resulting from dividing *d1* by *d2* . A
9 **Decimal** (the dividend). A **Decimal** (the divisor).

10 op_Multiply

11
12 [C#] public static decimal operator *(decimal d1, decimal d2);

13 [C++] public: static Decimal op_Multiply(Decimal d1, Decimal d2);

14 [VB] returnValue = Decimal.op_Multiply(d1, d2)

15 [JScript] returnValue = d1 * d2;

16
17 *Description*

18 Multiplies two specified **Decimal** values.

19 *Return Value:* The **Decimal** result of multiplying *d1* by *d2* . A **Decimal**. A
20 **Decimal**.

21 op_Subtraction

22
23 [C#] public static decimal operator -(decimal d1, decimal d2);

24 [C++] public: static Decimal op_Subtraction(Decimal d1, Decimal d2);

25 [VB] returnValue = Decimal.op_Subtraction(d1, d2)

1 [JScript] returnValue = d1 - d2;

3 *Description*

4 Subtracts two specified **Decimal** values.

5 *Return Value:* The **Decimal** result of subtracting *d2* from *d1* . A **Decimal**. A
6 **Decimal**.

7 op_UnaryNegation

9 [C#] public static decimal operator -(decimal d);

10 [C++] public: static Decimal op_UnaryNegation(Decimal d);

11 [VB] returnValue = Decimal.op_UnaryNegation(d)

12 [JScript] returnValue = -d;

14 *Description*

15 Negates the value of the **Decimal** operand.

16 *Return Value:* The negated value of the operand, *d* . The **Decimal** operand.

17 op_UnaryPlus

19 [C#] public static decimal operator +(decimal d);

20 [C++] public: static Decimal op_UnaryPlus(Decimal d);

21 [VB] returnValue = Decimal.op_UnaryPlus(d)

22 [JScript] returnValue = +d;

24 *Description*

1 Returns the value of the **Decimal** operand (the sign of the operand is
2 unchanged).

3 *Return Value:* The value of the operand, *d* . The **Decimal** operand.

4 Parse

5
6 [C#] public static decimal Parse(string s);

7 [C++] public: static Decimal Parse(String* s);

8 [VB] Public Shared Function Parse(ByVal s As String) As Decimal

9 [JScript] public static function Parse(s : String) : Decimal; Converts the **String**
10 representation of a number to its **Decimal** equivalent.

11 12 Description

13 Converts the **String** representation of a number to its **Decimal** equivalent.

14 *Return Value:* The **Decimal** number equivalent to the number contained in *s* .

15 *s* contains a number of the form: [ws][sign]digits[.fractional-digits][ws]

16 Items in square brackets ('[' and ']') are optional, and other items are as follows. A

17 **System.String** containing a number to convert.

18 Parse

19
20 [C#] public static decimal Parse(string s, IFormatProvider provider);

21 [C++] public: static Decimal Parse(String* s, IFormatProvider* provider);

22 [VB] Public Shared Function Parse(ByVal s As String, ByVal provider As
23 IFormatProvider) As Decimal

24 [JScript] public static function Parse(s : String, provider : IFormatProvider) :

25 Decimal;

Description

Converts the **String** representation of a number in a specified style to its **Decimal** equivalent.

Return Value: A **Decimal** with the value represented by *s* .

s contains a number of the form: [ws][sign]digits[.fractional-digits][ws]

Items in square brackets ('[' and ']') are optional, and other items are as follows. A

System.String containing a number to convert. An **System.IFormatProvider**

interface implementation which supplies culture-specific formatting information about *s*.

Parse

[C#] public static decimal Parse(string s, NumberStyles style);

[C++] public: static Decimal Parse(String* s, NumberStyles style);

[VB] Public Shared Function Parse(ByVal s As String, ByVal style As NumberStyles) As Decimal

[JScript] public static function Parse(s : String, style : NumberStyles) : Decimal;

Description

Converts the **String** representation of a number in a specified style to its **Decimal** equivalent.

Return Value: A **Decimal** with the value represented by *s* .

s contains a number of the form: [ws][sign]digits[.fractional-digits][ws]

Items in square brackets ('[' and ']') are optional, and other items are as follows. A

System.String containing a number to convert. The combination of one or more

System.Globalization.NumberStyles constants that indicate the permitted format of *s*.

Parse

[C#] public static decimal Parse(string s, NumberStyles style, IFormatProvider provider);

[C++] public: static Decimal Parse(String* s, NumberStyles style, IFormatProvider* provider);

[VB] Public Shared Function Parse(ByVal s As String, ByVal style As NumberStyles, ByVal provider As IFormatProvider) As Decimal

[JScript] public static function Parse(s : String, style : NumberStyles, provider : IFormatProvider) : Decimal;

Description

Converts the **String** representation of a number in a specified style and culture-specific format to its **Decimal** equivalent.

Return Value: A **Decimal** with the value represented by *s*.

s contains a number of the form: [ws][sign]digits[.fractional-digits][ws]

Items in square brackets ('[' and ']') are optional, and other items are as follows. A

System.String containing a number to convert. The combination of one or more **System.Globalization.NumberStyles** constants that indicate the permitted format of *s*. An **System.IFormatProvider** interface implementation which supplies culture-specific formatting information about *s*.

Remainder

```

1
2 [C#] public static decimal Remainder(decimal d1, decimal d2);
3 [C++] public: static Decimal Remainder(Decimal d1, Decimal d2);
4 [VB] Public Shared Function Remainder(ByVal d1 As Decimal, ByVal d2 As
5 Decimal) As Decimal
6 [JScript] public static function Remainder(d1 : Decimal, d2 : Decimal) : Decimal;
7

```

Description

Round

```

12 [C#] public static decimal Round(decimal d, int decimals);
13 [C++] public: static Decimal Round(Decimal d, int decimals);
14 [VB] Public Shared Function Round(ByVal d As Decimal, ByVal decimals As
15 Integer) As Decimal
16 [JScript] public static function Round(d : Decimal, decimals : int) : Decimal;
17

```

Description

Rounds a **Decimal** value to a specified number of decimal places.

Return Value: *d* rounded to *decimals* number of decimal places. A **Decimal** value to round. A value from 0 to 28 that specifies the number of decimal places to round to.

Subtract

```

25 [C#] public static decimal Subtract(decimal d1, decimal d2);

```

```

1 [C++] public: static Decimal Subtract(Decimal d1, Decimal d2);
2 [VB] Public Shared Function Subtract(ByVal d1 As Decimal, ByVal d2 As
3 Decimal) As Decimal
4 [JScript] public static function Subtract(d1 : Decimal, d2 : Decimal) : Decimal;
5

```

Description

Subtracts two specified **Decimal** values.

Return Value: The **Decimal** result of subtracting *d2* from *d1* . A **Decimal** (the minuend). A **Decimal** (the subtrahend).

Convertible.ToBoolean

```

12 [C#] bool IConvertible.ToBoolean(IFormatProvider provider);
13 [C++] bool IConvertible::ToBoolean(IFormatProvider* provider);
14 [VB] Function ToBoolean(ByVal provider As IFormatProvider) As Boolean
15 Implements IConvertible.ToBoolean
16 [JScript] function IConvertible.ToBoolean(provider : IFormatProvider) : Boolean;

```

Convertible.ToByte

```

19 [C#] byte IConvertible.ToByte(IFormatProvider provider);
20 [C++] unsigned char IConvertible::ToByte(IFormatProvider* provider);
21 [VB] Function ToByte(ByVal provider As IFormatProvider) As Byte Implements
22 IConvertible.ToByte
23 [JScript] function IConvertible.ToByte(provider : IFormatProvider) : Byte;

```

Convertible.ToChar

```

1
2 [C#] char IConvertible.ToChar(IFormatProvider provider);
3 [C++] __wchar_t IConvertible::ToChar(IFormatProvider* provider);
4 [VB] Function ToChar(ByVal provider As IFormatProvider) As Char Implements
5 IConvertible.ToChar
6 [JScript] function IConvertible.ToChar(provider : IFormatProvider) : Char;
7     IConvertible.ToDateTime
8
9 [C#] DateTime IConvertible.ToDateTime(IFormatProvider provider);
10 [C++] DateTime IConvertible::ToDateTime(IFormatProvider* provider);
11 [VB] Function ToDateTime(ByVal provider As IFormatProvider) As DateTime
12 Implements IConvertible.ToDateTime
13 [JScript] function IConvertible.ToDateTime(provider : IFormatProvider) :
14 DateTime;
15     IConvertible.ToDecimal
16
17 [C#] decimal IConvertible.ToDecimal(IFormatProvider provider);
18 [C++] Decimal IConvertible::ToDecimal(IFormatProvider* provider);
19 [VB] Function ToDecimal(ByVal provider As IFormatProvider) As Decimal
20 Implements IConvertible.ToDecimal
21 [JScript] function IConvertible.ToDecimal(provider : IFormatProvider) : Decimal;
22     IConvertible.ToDouble
23
24 [C#] double IConvertible.ToDouble(IFormatProvider provider);
25 [C++] double IConvertible::ToDouble(IFormatProvider* provider);

```

```

1  [VB] Function ToDouble(ByVal provider As IFormatProvider) As Double
2  Implements IConvertible.ToDouble
3  [JScript] function IConvertible.ToDouble(provider : IFormatProvider) : double;
4      IConvertible.ToInt16
5
6  [C#] short IConvertible.ToInt16(IFormatProvider provider);
7  [C++] short IConvertible::ToInt16(IFormatProvider* provider);
8  [VB] Function ToInt16(ByVal provider As IFormatProvider) As Short
9  Implements IConvertible.ToInt16
10 [JScript] function IConvertible.ToInt16(provider : IFormatProvider) : Int16;
11     IConvertible.ToInt32
12
13 [C#] int IConvertible.ToInt32(IFormatProvider provider);
14 [C++] int IConvertible::ToInt32(IFormatProvider* provider);
15 [VB] Function ToInt32(ByVal provider As IFormatProvider) As Integer
16 Implements IConvertible.ToInt32
17 [JScript] function IConvertible.ToInt32(provider : IFormatProvider) : int;
18     IConvertible.ToInt64
19
20 [C#] long IConvertible.ToInt64(IFormatProvider provider);
21 [C++] __int64 IConvertible::ToInt64(IFormatProvider* provider);
22 [VB] Function ToInt64(ByVal provider As IFormatProvider) As Long Implements
23 IConvertible.ToInt64
24 [JScript] function IConvertible.ToInt64(provider : IFormatProvider) : long;
25     IConvertible.ToSByte

```

```

1
2 [C#] sbyte IConvertible.ToSByte(IFormatProvider provider);
3 [C++] char IConvertible::ToSByte(IFormatProvider* provider);
4 [VB] Function ToSByte(ByVal provider As IFormatProvider) As SByte
5 Implements IConvertible.ToSByte
6 [JScript] function IConvertible.ToSByte(provider : IFormatProvider) : SByte;
7     IConvertible.ToSingle
8
9 [C#] float IConvertible.ToSingle(IFormatProvider provider);
10 [C++] float IConvertible::ToSingle(IFormatProvider* provider);
11 [VB] Function ToSingle(ByVal provider As IFormatProvider) As Single
12 Implements IConvertible.ToSingle
13 [JScript] function IConvertible.ToSingle(provider : IFormatProvider) : float;
14     IConvertible.ToType
15
16 [C#] object IConvertible.ToType(Type type, IFormatProvider provider);
17 [C++] Object* IConvertible::ToType(Type* type, IFormatProvider* provider);
18 [VB] Function ToType(ByVal type As Type, ByVal provider As IFormatProvider)
19 As Object Implements IConvertible.ToType
20 [JScript] function IConvertible.ToType(type : Type, provider : IFormatProvider) :
21 Object;
22     IConvertible.ToUInt16
23
24 [C#] ushort IConvertible.ToUInt16(IFormatProvider provider);
25 [C++] unsigned short IConvertible::ToUInt16(IFormatProvider* provider);

```



```

1  [VB] Function ToUInt16(ByVal provider As IFormatProvider) As UInt16
2  Implements IConvertible.ToUInt16
3  [JScript] function IConvertible.ToUInt16(provider : IFormatProvider) : UInt16;
4      IConvertible.ToUInt32
5
6  [C#] uint IConvertible.ToUInt32(IFormatProvider provider);
7  [C++] unsigned int IConvertible::ToUInt32(IFormatProvider* provider);
8  [VB] Function ToUInt32(ByVal provider As IFormatProvider) As UInt32
9  Implements IConvertible.ToUInt32
10 [JScript] function IConvertible.ToUInt32(provider : IFormatProvider) : UInt32;
11     IConvertible.ToUInt64
12
13 [C#] ulong IConvertible.ToUInt64(IFormatProvider provider);
14 [C++] unsigned __int64 IConvertible::ToUInt64(IFormatProvider* provider);
15 [VB] Function ToUInt64(ByVal provider As IFormatProvider) As UInt64
16 Implements IConvertible.ToUInt64
17 [JScript] function IConvertible.ToUInt64(provider : IFormatProvider) : UInt64;
18     ToByte
19
20 [C#] public static byte ToByte(decimal value);
21 [C++] public: static unsigned char ToByte(Decimal value);
22 [VB] Public Shared Function ToByte(ByVal value As Decimal) As Byte
23 [JScript] public static function ToByte(value : Decimal) : Byte;
24
25

```

Description

Converts the value of the specified **Decimal** to the equivalent 8-bit unsigned integer.

Return Value: An 8-bit unsigned integer equivalent to *value* .

value is rounded to the nearest integer value towards zero, and that result is returned. The **Decimal** value.

ToDouble

[C#] public static double ToDouble(decimal d);

[C++] public: static double ToDouble(Decimal d);

[VB] Public Shared Function ToDouble(ByVal d As Decimal) As Double

[JScript] public static function ToDouble(d : Decimal) : double;

Description

Converts the value of the specified **Decimal** to the equivalent double-precision floating point number.

Return Value: A double-precision floating point number equivalent to *d* .

Since a **Double** has fewer significant digits than a **Decimal** , this operation can produce round-off errors. The **Decimal** value to convert.

ToInt16

[C#] public static short ToInt16(decimal value);

[C++] public: static short ToInt16(Decimal value);

[VB] Public Shared Function ToInt16(ByVal value As Decimal) As Short

[JScript] public static function ToInt16(value : Decimal) : Int16;

Description

Converts the value of the specified **Decimal** to the equivalent 16-bit signed integer.

Return Value: An 16-bit signed integer equivalent to *value* . A **Decimal** value.

ToInt32

[C#] public static int ToInt32(decimal d);

[C++] public: static int ToInt32(Decimal d);

[VB] Public Shared Function ToInt32(ByVal d As Decimal) As Integer

[JScript] public static function ToInt32(d : Decimal) : int;

Description

Converts the value of the specified **Decimal** to the equivalent 32-bit signed integer.

Return Value: A 32-bit signed integer equivalent to the value of *d* .

The return value is the integral part of the decimal value; fractional digits are truncated. The **Decimal** value to convert.

ToInt64

[C#] public static long ToInt64(decimal d);

[C++] public: static __int64 ToInt64(Decimal d);

[VB] Public Shared Function ToInt64(ByVal d As Decimal) As Long

[JScript] public static function ToInt64(d : Decimal) : long;

Description

Converts the value of the specified **Decimal** to the equivalent 64-bit signed integer.

Return Value: A 64-bit signed integer equivalent to the value of *d*.

The return value is the integral part of the decimal value; fractional digits are truncated. The **Decimal** value to convert.

ToOACurrency

[C#] public static long ToOACurrency(decimal value);

[C++] public: static __int64 ToOACurrency(Decimal value);

[VB] Public Shared Function ToOACurrency(ByVal value As Decimal) As Long

[JScript] public static function ToOACurrency(value : Decimal) : long;

Description

Converts the specified **Decimal** value to the equivalent OLE Automation Currency value, which is contained in a 64-bit signed integer.

Return Value: A 64-bit signed integer that contains the OLE Automation equivalent of *value*. A **Decimal** value.

ToSByte

[C#] public static sbyte ToSByte(decimal value);

[C++] public: static char ToSByte(Decimal value);

[VB] Public Shared Function ToSByte(ByVal value As Decimal) As SByte

[JScript] public static function ToSByte(value : Decimal) : SByte;

Description

Converts the value of the specified **Decimal** to the equivalent 8-bit signed integer.

Return Value: An 8-bit signed integer equivalent to *value* . A **Decimal** value.

ToSingle

[C#] public static float ToSingle(decimal d);

[C++] public: static float ToSingle(Decimal d);

[VB] Public Shared Function ToSingle(ByVal d As Decimal) As Single

[JScript] public static function ToSingle(d : Decimal) : float;

Description

Converts the value of the specified **Decimal** to the equivalent single-precision floating point number.

Return Value: A single-precision floating point number equivalent to the value of *d* .

This operation can produce round-off errors because a single-precision floating point number has fewer significant digits than a **Decimal** . A **Decimal** value to convert.

ToString

[C#] public override string ToString();

[C++] public: String* ToString();

[VB] Overrides Public Function ToString() As String

[JScript] public override function ToString() : String; Converts the numeric value of this instance to its equivalent **String** representation.

Description

Converts the numeric value of this instance to its equivalent **String** representation.

Return Value: A **String** representing the value of this instance.

The return value is formatted with the general format specifier ("G"). That is, an optional minus sign symbol followed by a sequence of integral digits ("0" through "9"), optionally followed by a decimal point symbol and a sequence of fractional digits. No leading zeroes are prefixed to the returned value.

ToString

[C#] public string ToString(IFormatProvider provider);

[C++] public: __sealed String* ToString(IFormatProvider* provider);

[VB] NotOverridable Public Function ToString(ByVal provider As IFormatProvider) As String

[JScript] public function ToString(provider : IFormatProvider) : String;

Description

Converts the numeric value of this instance to its equivalent **String** representation using the specified culture-specific format information.

Return Value: The **String** representation of the value of this instance as specified by *provider* .

provider is an **IFormatProvider** instance that obtains a **System.Globalization.NumberFormatInfo** object. The **NumberFormatInfo** object provides culture-specific format information about this instance. If *provider* is **null**, the return value for this instance is formatted with the **NumberFormatInfo** for the current culture. An **System.IFormatProvider** interface implementation which supplies culture-specific formatting information.

ToString

```
[C#] public string ToString(string format);
[C++] public: String* ToString(String* format);
[VB] Public Function ToString(ByVal format As String) As String
[JScript] public function ToString(format : String) : String;
```

Description

Converts the numeric value of this instance to its equivalent **String** representation, using the specified format specification.

Return Value: A **String** representation of the value of this instance as specified by *format*.

If *format* is **null** or an empty string, the return value of this instance is formatted with the general format specifier ("G"). A **String** containing a format specification.

ToString

```
[C#] public string ToString(string format, IFormatProvider provider);
[C++] public: __sealed String* ToString(String* format, IFormatProvider*
```

provider);

[VB] NotOverridable Public Function ToString(ByVal format As String, ByVal provider As IFormatProvider) As String

[JScript] public function ToString(format : String, provider : IFormatProvider) : String;

Description

Converts the numeric value of this instance to its equivalent **String** representation using the specified format and culture-specific format information.

Return Value: The **String** representation of the value of this instance as specified by *format* and *provider* .

If *format* is **null** or an empty string, the return value for this instance is formatted with the general format specifier ("G"). A format specification. An **System.IFormatProvider** interface implementation which supplies culture-specific formatting information.

ToUInt16

[C#] public static ushort ToUInt16(decimal value);

[C++] public: static unsigned short ToUInt16(Decimal value);

[VB] Public Shared Function ToUInt16(ByVal value As Decimal) As UInt16

[JScript] public static function ToUInt16(value : Decimal) : UInt16;

Description

Converts the value of the specified **Decimal** to the equivalent 16-bit unsigned integer.

Return Value: A 16-bit unsigned integer equivalent to the value of *value* .

The return value is the integral part of the decimal value; fractional digits are truncated. A **Decimal** value to convert.

ToUInt32

[C#] public static uint ToUInt32(decimal d);

[C++] public: static unsigned int ToUInt32(Decimal d);

[VB] Public Shared Function ToUInt32(ByVal d As Decimal) As UInt32

[JScript] public static function ToUInt32(d : Decimal) : UInt32;

Description

Converts the value of the specified **Decimal** to the equivalent 32-bit unsigned integer.

Return Value: A 32-bit unsigned integer equivalent to the value of *d* .

The return value is the integral part of the decimal value; fractional digits are truncated. A **Decimal** value to convert.

ToUInt64

[C#] public static ulong ToUInt64(decimal d);

[C++] public: static unsigned __int64 ToUInt64(Decimal d);

[VB] Public Shared Function ToUInt64(ByVal d As Decimal) As UInt64

[JScript] public static function ToUInt64(d : Decimal) : UInt64;

1
2 *Description*

3 Converts the value of the specified **Decimal** to the equivalent 64-bit
4 unsigned integer.

5 *Return Value:* A 64-bit unsigned integer equivalent to the value of *d* .

6 The return value is the integral part of the decimal value; fractional digits
7 are truncated. A **Decimal** value to convert.

8 Truncate

9
10 [C#] public static decimal Truncate(decimal d);

11 [C++] public: static Decimal Truncate(Decimal d);

12 [VB] Public Shared Function Truncate(ByVal d As Decimal) As Decimal

13 [JScript] public static function Truncate(d : Decimal) : Decimal;

14
15 *Description*

16 Returns the integral digits of the specified **Decimal** ; any fractional digits
17 are discarded.

18 *Return Value:* *d* rounded toward zero, to the nearest whole number.

19 This method rounds parameter *d* towards zero to the nearest whole number,
20 which corresponds to discarding any digits after the decimal point. A **Decimal** to
21 truncate.

22 Delegate class (System)

23 Truncate

Description

Represents a delegate, which is a data structure that refers to a static method or to an object instance and an instance method of that object.

The **System.Delegate** class is the base class for delegates.

Delegate

Example Syntax:

Truncate

[C#] protected Delegate(object target, string method);

[C++] protected: Delegate(Object* target, String* method);

[VB] Protected Sub New(ByVal target As Object, ByVal method As String)

[JScript] protected function Delegate(target : Object, method : String); Initializes a new instance of the **System.Delegate** class.

Description

Creates a **System.Delegate** to represent the specified target **System.Object** and the specified method.

This protected constructor is accessible only through this class or a derived class. The **System.Object** to be represented by the **System.Delegate**. The **System.String** containing the name of the method to be represented by the **System.Delegate**.

Delegate

Example Syntax:

1 Truncate

2

3 [C#] protected Delegate(Type target, string method);

4 [C++] protected: Delegate(Type* target, String* method);

5 [VB] Protected Sub New(ByVal target As Type, ByVal method As String)

6 [JScript] protected function Delegate(target : Type, method : String);

7

8 *Description*

9 Creates a **System.Delegate** to represent the specified target **System.Type**
10 and the specified method.

11 This protected constructor is accessible only through this class or a derived
12 class. The **System.Type** to be represented by the **System.Delegate**. The
13 **System.String** containing the name of the method to be represented by the
14 **System.Delegate**.

15 Method

16 Truncate

17

18 [C#] public MethodInfo Method {get;}

19 [C++] public: __property MethodInfo* get_Method();

20 [VB] Public ReadOnly Property Method As MethodInfo

21 [JScript] public function get Method() : MethodInfo;

22

23 *Description*

24 Gets the static method of the class represented by the **System.Delegate** .

25 Target

Truncate

```
[C#] public object Target {get;}
[C++] public: __property Object* get_Target();
[VB] Public ReadOnly Property Target As Object
[JScript] public function get Target() : Object;
```

Description

Gets the class instance from which the **System.Delegate** was created.

An instance method is a method that is associated with an instance of a class; whereas, a static method is a method that is associated with the class itself.

Clone

```
[C#] public virtual object Clone();
[C++] public: virtual Object* Clone();
[VB] Overridable Public Function Clone() As Object
[JScript] public function Clone() : Object;
```

Description

Creates a shallow copy of the **System.Delegate** .

Return Value: A shallow copy of the **System.Delegate** .

This method can be overridden by a derived class.

Combine

```
[C#] public static Delegate Combine(Delegate[] delegates);
```

```

1 [C++] public: static Delegate* Combine(Delegate* delegates[]);
2 [VB] Public Shared Function Combine(ByVal delegates() As Delegate) As
3 Delegate
4 [JScript] public static function Combine(delegates : Delegate[]) : Delegate;
5

```

Description

Combines the invocation lists of an array of multicast **System.Delegate** instances.

Return Value: A new multicast **System.Delegate** with an invocation list that concatenates the invocation lists of the delegates in the *delegates* array.

If the *delegates* array contains entries that are **null**, those entries are ignored. The array of multicast **System.Delegate** instances to combine.

Combine

```

15 [C#] public static Delegate Combine(Delegate a, Delegate b);
16 [C++] public: static Delegate* Combine(Delegate* a, Delegate* b);
17 [VB] Public Shared Function Combine(ByVal a As Delegate, ByVal b As
18 Delegate) As Delegate
19 [JScript] public static function Combine(a : Delegate, b : Delegate) : Delegate;
20 Combines the invocation lists of the specified multicast System.Delegate
21 instances.
22

```

Description

Combines the invocation lists of two multicast **System.Delegate** instances.

Return Value: A new multicast **System.Delegate** with an invocation list that concatenates the invocation lists of *a* and *b* in that order.

The invocation list can contain duplicate entries; that is, entries that refer to the same method on the same object. The multicast **System.Delegate** whose invocation list comes first. The multicast **System.Delegate** whose invocation list comes last.

CombineImpl

[C#] protected virtual Delegate CombineImpl(Delegate d);

[C++] protected: virtual Delegate* CombineImpl(Delegate* d);

[VB] Overridable Protected Function CombineImpl(ByVal d As Delegate) As Delegate

[JScript] protected function CombineImpl(d : Delegate) : Delegate;

Description

When overridden in a derived class, combines the invocation lists of the specified multicast **System.Delegate** with the current multicast **System.Delegate**.

Return Value: When overridden in a derived class, a new **System.Delegate** with an invocation list that concatenates the invocation list of the current **System.Delegate** and the invocation list of *d*.

This method must be overridden by a derived class. The current implementation simply throws a **System.MulticastNotSupportedException**.

This protected method is accessible only through this class or a derived class. The

multicast **System.Delegate** whose invocation list to append to the end of the invocation list of the current multicast **System.Delegate**.

CreateDelegate

[C#] public static Delegate CreateDelegate(Type type, MethodInfo method);
[C++] public: static Delegate* CreateDelegate(Type* type, MethodInfo* method);
[VB] Public Shared Function CreateDelegate(ByVal type As Type, ByVal method As MethodInfo) As Delegate
[JScript] public static function CreateDelegate(type : Type, method : MethodInfo) : Delegate;

Description

Creates a **System.Delegate** of the specified type to represent the specified static method.

Return Value: A **System.Delegate** of the specified type to represent the specified static method.

This method creates delegates for static methods only. A static method is a method that is associated with the class itself. The **System.Type** of **System.Delegate** to create. The **System.Reflection.MethodInfo** describing the static method for which the **System.Delegate** is to be created.

CreateDelegate

[C#] public static Delegate CreateDelegate(Type type, object target, string method);
[C++] public: static Delegate* CreateDelegate(Type* type, Object* target, String*

method);

[VB] Public Shared Function CreateDelegate(ByVal type As Type, ByVal target
As Object, ByVal method As String) As Delegate

[JScript] public static function CreateDelegate(type : Type, target : Object, method
: String) : Delegate; Creates a **System.Delegate** of the specified **System.Type** .

Description

Creates a **System.Delegate** of the specified type to represent the specified
instance method of the specified **System.Object** instance.

Return Value: A **System.Delegate** of the specified type to represent the specified
instance method of the specified **System.Object** instance *type* is **null** .

This method creates delegates for instance methods only. An instance
method is a method that is associated with an instance of a class. The
System.Type of **System.Delegate** to create. The target **System.Object** instance
that implements *method*. The name of the instance method for which the
System.Delegate is to be created.

CreateDelegate

[C#] public static Delegate CreateDelegate(Type type, Type target, string
method);

[C++] public: static Delegate* CreateDelegate(Type* type, Type* target, String*
method);

[VB] Public Shared Function CreateDelegate(ByVal type As Type, ByVal target
As Type, ByVal method As String) As Delegate

[JScript] public static function CreateDelegate(type : Type, target : Type, method :

String) : Delegate;

Description

Creates a **System.Delegate** of the specified type to represent the specified static method of the specified **System.Type**.

Return Value: A **System.Delegate** of the specified type to represent the specified static method of the specified **System.Type**.

This method creates delegates for static methods only. A static method is a method that is associated with the class itself, not with any particular instance of the class. The **System.Type** of **System.Delegate** to create. The target **System.Type** that implements *method*. The name of the static method for which the delegate is to be created.

DynamicInvoke

[C#] public object DynamicInvoke(object[] args);

[C++] public: Object* DynamicInvoke(Object* args __gc[]);

[VB] Public Function DynamicInvoke(ByVal args() As Object) As Object

[JScript] public function DynamicInvoke(args : Object[]) : Object;

Description

Invokes the method, represented by the **System.Delegate**, dynamically (late-bound).

Return Value: The **System.Object** returned by the method represented by the **System.Delegate**.

This method calls the **System.Delegate.DynamicInvokeImpl(System.Object[])** method. An array of **System.Object** instances that are the arguments to pass to the method represented by the **System.Delegate**.

DynamicInvokeImpl

[C#] protected virtual object DynamicInvokeImpl(object[] args);

[C++] protected: virtual Object* DynamicInvokeImpl(Object* args __gc[]);

[VB] Overridable Protected Function DynamicInvokeImpl(ByVal args() As Object) As Object

[JScript] protected function DynamicInvokeImpl(args : Object[]) : Object;

Description

Invokes the method, represented by the **System.Delegate** , dynamically (late-bound).

Return Value: The **System.Object** returned by the method represented by the **System.Delegate** .

This method can be overridden by a derived class. This protected method is accessible only through this class or a derived class. An array of **System.Object** instances that are the arguments to pass to the method represented by the **System.Delegate**.

Equals

[C#] public override bool Equals(object obj);

[C++] public: bool Equals(Object* obj);

1 [VB] Overrides Public Function Equals(ByVal obj As Object) As Boolean

2 [JScript] public override function Equals(obj : Object) : Boolean;

3
4 *Description*

5 Determines whether the specified object and the singlecast

6 **System.Delegate** share the same target, method and invocation list.

7 *Return Value:* **true** if *obj* and the current **System.Delegate** have the same target,
8 method and invocation list; otherwise, **false** .

9 Two delegates with the same methods, the same targets and the same
10 invocation lists are considered equal, even if they are not both singlecast or both
11 multicast. The **System.Object** to compare with the singlecast **System.Delegate**.

12 GetHashCode

13
14 [C#] public override int GetHashCode();

15 [C++] public: int GetHashCode();

16 [VB] Overrides Public Function GetHashCode() As Integer

17 [JScript] public override function GetHashCode() : int;

18
19 *Description*

20 Returns a hash code for the delegate instance.

21 *Return Value:* A hash code for the delegate instance.

22 The return value from this method must not be persisted for two reasons.
23 First, the hash function of a class might be altered to generate a better distribution,
24 rendering any values from the old hash function useless. Second, the default
25

implementation of this class does not guarantee that the same value will be returned by different instances.

GetInvocationList

[C#] public virtual Delegate[] GetInvocationList();

[C++] public: virtual Delegate* GetInvocationList() [];

[VB] Overridable Public Function GetInvocationList() As Delegate()

[JScript] public function GetInvocationList() : Delegate[];

Description

Returns the invocation list of the **System.Delegate**.

Return Value: An array of singlecast **System.Delegate** objects representing the invocation list of the current **System.Delegate**. If the current **System.Delegate** is singlecast, the array contains only one element. If the current **System.Delegate** is multicast, the array may contain more than one element.

This method can be overridden by a derived class.

GetMethodImpl

[C#] protected virtual MethodInfo GetMethodImpl();

[C++] protected: virtual MethodInfo* GetMethodImpl();

[VB] Overridable Protected Function GetMethodImpl() As MethodInfo

[JScript] protected function GetMethodImpl() : MethodInfo;

Description

Gets the static method of the class represented by the **System.Delegate** .

Return Value: A **System.Reflection.MethodInfo** describing the static method represented by the **System.Delegate** .

This method can be overridden by a derived class. This protected method is accessible only through this class or a derived class.

GetObjectData

[C#] public virtual void GetObjectData(SerializationInfo info, StreamingContext context);

[C++] public: virtual void GetObjectData(SerializationInfo* info, StreamingContext context);

[VB] Overridable Public Sub GetObjectData(ByVal info As SerializationInfo, ByVal context As StreamingContext)

[JScript] public function GetObjectData(info : SerializationInfo, context : StreamingContext);

Description

Implements the **System.Runtime.Serialization.ISerializable** interface and returns the data needed to serialize the **System.Delegate** .

This method can be overridden by a derived class. A **System.Runtime.Serialization.SerializationInfo** object containing information required to serialize the **System.Delegate**. A **System.Runtime.Serialization.StreamingContext** object containing the source and destination of the serialized stream associated with the **System.Delegate**.

op_Equality

```

1
2 [C#] public static bool operator ==(Delegate d1, Delegate d2);
3 [C++] public: static bool op_Equality(Delegate* d1, Delegate* d2);
4 [VB] returnValue = Delegate.op_Equality(d1, d2)
5 [JScript] returnValue = d1 == d2;
6

```

Description

Determines whether the specified **System.Delegate** objects are equal.

Return Value: **true** if *d1* is equal to *d2* ; otherwise, **false** .

Two delegates with the same methods, the same targets and the same invocation lists are considered equal, even if they are not both singlecast or both multicast. The first **System.Delegate** to compare. The second **System.Delegate** to compare.

op_Inequality

```

14
15
16 [C#] public static bool operator !=(Delegate d1, Delegate d2);
17 [C++] public: static bool op_Inequality(Delegate* d1, Delegate* d2);
18 [VB] returnValue = Delegate.op_Inequality(d1, d2)
19 [JScript] returnValue = d1 != d2;
20

```

Description

Determines whether the specified **System.Delegate** objects are not equal.

Return Value: **true** if *d1* is not equal to *d2* ; otherwise, **false** .

Two delegates are considered not equal if they have different methods or different targets or different invocation lists. The first **System.Delegate** to compare. The second **System.Delegate** to compare.

Remove

```
[C#] public static Delegate Remove(Delegate source, Delegate value);  
[C++] public: static Delegate* Remove(Delegate* source, Delegate* value);  
[VB] Public Shared Function Remove(ByVal source As Delegate, ByVal value As  
Delegate) As Delegate  
[JScript] public static function Remove(source : Delegate, value : Delegate) :  
Delegate;
```

Description

Removes a **System.Delegate** from the invocation list of another **System.Delegate**.

Return Value: A new **System.Delegate** with an invocation list formed by taking the invocation list of source and removing the last occurrence of *value*, if *value* is found in the invocation list of **source**.

The **System.Delegate** removed from the invocation list is the last delegate for which the following expression is true: *value*. The **System.Delegate** from which to remove *value*. The **System.Delegate** to remove from the invocation list of *source*.

RemoveImpl

```
[C#] protected virtual Delegate RemoveImpl(Delegate d);
```


1 [C++] protected: virtual Delegate* RemoveImpl(Delegate* d);

2 [VB] Overridable Protected Function RemoveImpl(ByVal d As Delegate) As

3 Delegate

4 [JScript] protected function RemoveImpl(d : Delegate) : Delegate;

5
6 *Description*

7 Removes a **System.Delegate** from the invocation list of another

8 **System.Delegate** .

9 *Return Value:* *source* , if *d* is not equal to *source* ; otherwise, **null** .

10 This method can be overridden by a derived class. This protected method is
11 accessible only through this class or a derived class. The **System.Delegate** to
12 remove from the invocation list of the current **System.Delegate**.

13 DivideByZeroException class (System)

14 ToString

15
16
17 *Description*

18 The exception that is thrown when there is an attempt to divide an integral
19 or decimal value by zero.

20 Dividing a floating-point value by zero will result in either positive infinity,
21 negative infinity, or Not-a-Number (NaN) according to the rules of IEEE 754
22 arithmetic. Floating-point operations never throw an exception. For more
23 information, see **System.Single** and **System.Double** .

24 DivideByZeroException

25 *Example Syntax:*

ToString

[C#] public DivideByZeroException();

[C++] public: DivideByZeroException();

[VB] Public Sub New()

[JScript] public function DivideByZeroException(); Initializes a new instance of the **System.DivideByZeroException** class.

Description

Initializes a new instance of the **System.DivideByZeroException** class with default properties.

The following table shows the initial property values for an instance of **System.DivideByZeroException**.

DivideByZeroException

Example Syntax:

ToString

[C#] public DivideByZeroException(string message);

[C++] public: DivideByZeroException(String* message);

[VB] Public Sub New(ByVal message As String)

[JScript] public function DivideByZeroException(message : String);

Description

Initializes a new instance of the **System.DivideByZeroException** class with a specified error message.

The following table shows the initial property values for an instance of **System.DivideByZeroException**. The error message that explains the reason for the exception.

DivideByZeroException

Example Syntax:

ToString

[C#] protected DivideByZeroException(SerializationInfo info, StreamingContext context);

[C++] protected: DivideByZeroException(SerializationInfo* info, StreamingContext context);

[VB] Protected Sub New(ByVal info As SerializationInfo, ByVal context As StreamingContext)

[JScript] protected function DivideByZeroException(info : SerializationInfo, context : StreamingContext);

Description

Initializes a new instance of the **System.DivideByZeroException** class with serialized data.

This constructor is called during deserialization to reconstitute the exception object transmitted over a stream. For more information, see . The object that holds the serialized object data. The contextual information about the source or destination.

DivideByZeroException

Example Syntax:

ToString

```
[C#] public DivideByZeroException(string message, Exception innerException);  
[C++] public: DivideByZeroException(String* message, Exception*  
innerException);  
[VB] Public Sub New(ByVal message As String, ByVal innerException As  
Exception)  
[JScript] public function DivideByZeroException(message : String,  
innerException : Exception);
```

Description

Initializes a new instance of the **System.DivideByZeroException** class with a specified error message and a reference to the inner exception that is the root cause of this exception.

When an **Exception** *X* is thrown as a direct result of a previous exception *Y*, the **System.Exception.InnerException** property of *X* should contain a reference to *Y*. The **InnerException** property returns the same value as was passed into the constructor, or **null** if the inner exception value was not supplied to the constructor. The error message that explains the reason for the exception. An instance of **System.Exception** that is the cause of the current **Exception**. If *innerException* is non-null, then the current **Exception** is raised in a catch block handling *innerException*.

HelpLink

HResult

InnerException

1 Message

2 Source

3 StackTrace

4 TargetSite

5 DllNotFoundException class (System)

6 ToString

7
8
9 *Description*

10 The exception that is thrown when a DLL specified in a DLL import cannot
11 be found.

12 **System.DllNotFoundException** uses the HRESULT
13 COR_E_DLLNOTFOUND, which has the value 0x80131524.

14 DllNotFoundException

15 *Example Syntax:*

16 ToString

17
18 [C#] public DllNotFoundException();

19 [C++] public: DllNotFoundException();

20 [VB] Public Sub New()

21 [JScript] public function DllNotFoundException(); Initializes a new instance of
22 the **System.DllNotFoundException** class.

23
24 *Description*

1 Initializes a new instance of the **System.DllNotFoundException** class with
2 default properties.

3 The following table shows the initial property values for an instance of
4 **System.DllNotFoundException** .

5 DllNotFoundException

6 *Example Syntax:*

7 ToString

9 [C#] public DllNotFoundException(string message);

10 [C++] public: DllNotFoundException(String* message);

11 [VB] Public Sub New(ByVal message As String)

12 [JScript] public function DllNotFoundException(message : String);

14 *Description*

15 Initializes a new instance of the **System.DllNotFoundException** class with
16 a specified error message.

17 The following table shows the initial property values for an instance of
18 **System.DllNotFoundException** . The error message that explains the reason for
19 the exception.

20 DllNotFoundException

21 *Example Syntax:*

22 ToString

24 [C#] protected DllNotFoundException(SerializationInfo info, StreamingContext
25 context);

1 [C++] protected: DllNotFoundException(SerializationInfo* info,
2 StreamingContext context);

3 [VB] Protected Sub New(ByVal info As SerializationInfo, ByVal context As
4 StreamingContext)

5 [JScript] protected function DllNotFoundException(info : SerializationInfo,
6 context : StreamingContext);

8 *Description*

9 Initializes a new instance of the **System.DllNotFoundException** class with
10 serialized data. The **System.Runtime.Serialization.SerializationInfo** that holds
11 the serialized object data about the exception being thrown. The
12 **System.Runtime.Serialization.StreamingContext** that contains contextual
13 information about the source or destination.

14 DllNotFoundException

15 *Example Syntax:*

16 ToString

17
18 [C#] public DllNotFoundException(string message, Exception inner);

19 [C++] public: DllNotFoundException(String* message, Exception* inner);

20 [VB] Public Sub New(ByVal message As String, ByVal inner As Exception)

21 [JScript] public function DllNotFoundException(message : String, inner :
22 Exception);

23 24 *Description*

1 Initializes a new instance of the **System.DllNotFoundException** class with
2 a specified error message and a reference to the inner exception that is the root
3 cause of this exception.

4 When an **Exception** *X* is thrown as a direct result of a previous exception *Y* ,
5 the **System.Exception.InnerException** property of *X* should contain a reference
6 to *Y* . The **InnerException** property returns the same value as was passed into the
7 constructor, or **null** if the inner exception value was not supplied to the
8 constructor. The error message that explains the reason for the exception. An
9 instance of **System.Exception** that is the cause of the current **Exception**. If *inner*
10 is non-null, then the current **Exception** is raised in a catch block handling *inner* .

11 HelpLink

12 HResult

13 InnerException

14 Message

15 Source

16 StackTrace

17 TargetSite

18 TypeName

19 Double structure (System)

20 ToString

21
22
23 *Description*

24 Represents a double-precision floating point number.
25

The **Double** value type represents a double-precision 64-bit number with values ranging from negative 1.79769313486232e308 to positive 1.79769313486232e308, as well as positive or negative zero, **System.Double.PositiveInfinity** , **System.Double.NegativeInfinity** , and Not-a-Number (**System.Double.NaN**).

ToString

[C#] public const double Epsilon;

[C++] public: const double Epsilon;

[VB] Public Const Epsilon As Double

[JScript] public var Epsilon : double;

Description

A constant representing the smallest positive **Double** greater than zero.

The value of this constant is 4.94065645841247e-324.

ToString

[C#] public const double MaxValue;

[C++] public: const double MaxValue;

[VB] Public Const MaxValue As Double

[JScript] public var MaxValue : double;

Description

A constant representing the largest possible value of **Double** .

The value of this constant is positive 1.79769313486232e308.

ToString

```
[C#] public const double MinValue;  
[C++] public: const double MinValue;  
[VB] Public Const MinValue As Double  
[JScript] public var MinValue : double;
```

Description

A constant representing the smallest possible value of **Double** .

The value of this constant is negative 1.79769313486232e308.

ToString

```
[C#] public const double NaN;  
[C++] public: const double NaN;  
[VB] Public Const NaN As Double  
[JScript] public var NaN : double;
```

Description

A constant representing Not-a-Number (**NaN**).

The value of this constant is the result of dividing zero by zero.

ToString

```
[C#] public const double NegativeInfinity;  
[C++] public: const double NegativeInfinity;  
[VB] Public Const NegativeInfinity As Double
```

1 [JScript] public var NegativeInfinity : double;

3 *Description*

4 A constant representing negative infinity.

5 The value of this constant is the result of dividing a negative number by
6 zero.

7 ToString

9 [C#] public const double PositiveInfinity;

10 [C++] public: const double PositiveInfinity;

11 [VB] Public Const PositiveInfinity As Double

12 [JScript] public var PositiveInfinity : double;

14 *Description*

15 A constant representing positive infinity.

16 The value of this constant is the result of dividing a positive number by
17 zero.

18 CompareTo

20 [C#] public int CompareTo(object value);

21 [C++] public: __sealed int CompareTo(Object* value);

22 [VB] NotOverridable Public Function CompareTo(ByVal value As Object) As
23 Integer

24 [JScript] public function CompareTo(value : Object) : int;

Description

Compares this instance to a specified object and returns an indication of their relative values.

Return Value: A signed number indicating the relative values of this instance and *value*.

Any instance of **Double**, regardless of its value, is considered greater than **null**. An object to compare, or **null**.

Equals

[C#] public override bool Equals(object obj);

[C++] public: bool Equals(Object* obj);

[VB] Overrides Public Function Equals(ByVal obj As Object) As Boolean

[JScript] public override function Equals(obj : Object) : Boolean;

Description

Returns a value indicating whether this instance is equal to a specified object.

Return Value: **true** if *obj* is an instance of **Double** and equals the value of this instance; otherwise, **false**. An object to compare with this instance.

GetHashCode

[C#] public override int GetHashCode();

[C++] public: int GetHashCode();

[VB] Overrides Public Function GetHashCode() As Integer

1 [JScript] public override function GetHashCode() : int;

3 *Description*

4 Returns the hash code for this instance.

5 *Return Value:* A 32-bit signed integer hash code.

6 GetTypeCode

8 [C#] public TypeCode GetTypeCode();

9 [C++] public: __sealed TypeCode GetTypeCode();

10 [VB] NotOverridable Public Function GetTypeCode() As TypeCode

11 [JScript] public function GetTypeCode() : TypeCode;

13 *Description*

14 Returns the **TypeCode** for value type **Double** .

15 *Return Value:* The enumerated constant, **System.TypeCode.Double** .

16 IsInfinity

18 [C#] public static bool IsInfinity(double d);

19 [C++] public: static bool IsInfinity(double d);

20 [VB] Public Shared Function IsInfinity(ByVal d As Double) As Boolean

21 [JScript] public static function IsInfinity(d : double) : Boolean;

23 *Description*

24 Returns a value indicating whether the specified number evaluates to either
25 negative or positive infinity.

1 *Return Value:* **true** if *d* evaluates to negative or positive infinity; otherwise, **false** .

2 A double-precision floating point number.

3 NaN

4
5 [C#] public static bool IsNaN(double d);

6 [C++] public: static bool IsNaN(double d);

7 [VB] Public Shared Function IsNaN(ByVal d As Double) As Boolean

8 [JScript] public static function IsNaN(d : double) : Boolean;

9
10 *Description*

11 Returns a value indicating whether the specified number evaluates to Not-a-
12 Number (NaN).

13 *Return Value:* **true** if *d* evaluates to NaN; otherwise, **false** . A double-precision
14 floating point number.

15 IsNegativeInfinity

16
17 [C#] public static bool IsNegativeInfinity(double d);

18 [C++] public: static bool IsNegativeInfinity(double d);

19 [VB] Public Shared Function IsNegativeInfinity(ByVal d As Double) As Boolean

20 [JScript] public static function IsNegativeInfinity(d : double) : Boolean;

21
22 *Description*

23 Returns a value indicating whether the specified number evaluates to
24 negative infinity.

Return Value: **true** if *d* evaluates to negative infinity; otherwise, **false** . A double-precision floating point number.

IsPositiveInfinity

[C#] public static bool IsPositiveInfinity(double d);

[C++] public: static bool IsPositiveInfinity(double d);

[VB] Public Shared Function IsPositiveInfinity(ByVal d As Double) As Boolean

[JScript] public static function IsPositiveInfinity(d : double) : Boolean;

Description

Returns a value indicating whether the specified number evaluates to positive infinity.

Return Value: **true** if *d* evaluates to positive infinity; otherwise, **false** . A double-precision floating point number.

Parse

[C#] public static double Parse(string s);

[C++] public: static double Parse(String* s);

[VB] Public Shared Function Parse(ByVal s As String) As Double

[JScript] public static function Parse(s : String) : double; Converts the **String** representation of a number to its double-precision floating point number equivalent.

Description

Converts the **String** representation of a number to its double-precision floating point number equivalent.

Return Value: A double-precision floating point number equivalent to the numeric value or symbol specified in *s* .

s can contain

System.Globalization.NumberFormatInfo.PositiveInfinitySymbol ,
System.Globalization.NumberFormatInfo.NegativeInfinitySymbol ,
System.Globalization.NumberFormatInfo.NaNSymbol , or a string of the form:

[ws][sign]integral-digits[.[fractional-digits]][e[sign]exponential-digits][ws]

Optional items are framed in square brackets ('[' and ']'). Items containing the term "digits" consist of a series of numeric characters ranging from 0 to 9. A

System.String containing a number to convert.

Parse

[C#] public static double Parse(string s, IFormatProvider provider);

[C++] public: static double Parse(String* s, IFormatProvider* provider);

[VB] Public Shared Function Parse(ByVal s As String, ByVal provider As IFormatProvider) As Double

[JScript] public static function Parse(s : String, provider : IFormatProvider) : double;

Description

Converts the **String** representation of a number in a specified culture-specific format to its double-precision floating point number equivalent.

Return Value: A double-precision floating point number equivalent to the numeric value or symbol specified in *s* .

s can contain

System.Globalization.NumberFormatInfo.PositiveInfinitySymbol ,
System.Globalization.NumberFormatInfo.NegativeInfinitySymbol ,
System.Globalization.NumberFormatInfo.NaNSymbol , or a string of the form:

[ws][sign]integral-digits[.[fractional-digits]][e[sign]exponential-digits][ws]

Optional items are framed in square brackets ('[' and ']'). Items containing the term "digits" consist of a series of numeric characters ranging from 0 to 9. A

System.String containing a number to convert. An **System.IFormatProvider** interface implementation which supplies culture-specific formatting information about *s*.

Parse

[C#] public static double Parse(string s, NumberStyles style);

[C++] public: static double Parse(String* s, NumberStyles style);

[VB] Public Shared Function Parse(ByVal s As String, ByVal style As NumberStyles) As Double

[JScript] public static function Parse(s : String, style : NumberStyles) : double;

Description

Converts the **String** representation of a number in a specified style to its double-precision floating point number equivalent.

Return Value: A double-precision floating point number equivalent to the numeric value or symbol specified in *s* .

s can contain

System.Globalization.NumberFormatInfo.PositiveInfinitySymbol ,
System.Globalization.NumberFormatInfo.NegativeInfinitySymbol ,
System.Globalization.NumberFormatInfo.NaNSymbol , or a string of the form:
[ws][sign]integral-digits[.[fractional-digits]][e[sign]exponential-digits][ws]
Optional items are framed in square brackets ('[' and ']'). Items containing the term
"digits" consist of a series of numeric characters ranging from 0 to 9. A
System.String containing a number to convert. The combination of one or more
System.Globalization.NumberStyles constants that indicate the permitted format
of *s*.

Parse

[C#] public static double Parse(string s, NumberStyles style, IFormatProvider
provider);
[C++] public: static double Parse(String* s, NumberStyles style, IFormatProvider*
provider);
[VB] Public Shared Function Parse(ByVal s As String, ByVal style As
NumberStyles, ByVal provider As IFormatProvider) As Double
[JScript] public static function Parse(s : String, style : NumberStyles, provider :
IFormatProvider) : double;

Description

Converts the **String** representation of a number in a specified style and
culture-specific format to its double-precision floating point number equivalent.

Return Value: A double-precision floating point number equivalent to the numeric value or symbol specified in *s* .

s can contain

System.Globalization.NumberFormatInfo.PositiveInfinitySymbol ,
System.Globalization.NumberFormatInfo.NegativeInfinitySymbol ,
System.Globalization.NumberFormatInfo.NaNSymbol , or a string of the form:

[ws][sign]integral-digits[.[fractional-digits]][e[sign]exponential-digits][ws]

Optional items are framed in square brackets ('[' and ']'). Items containing the term "digits" consist of a series of numeric characters ranging from 0 to 9. A

System.String containing a number to convert. The combination of one or more **System.Globalization.NumberStyles** constants that indicate the permitted format of *s*. An **System.IFormatProvider** interface implementation which supplies culture-specific formatting information about *s*.

Convertible.ToBoolean

[C#] bool Convertible.ToBoolean(IFormatProvider provider);

[C++] bool Convertible::ToBoolean(IFormatProvider* provider);

[VB] Function ToBoolean(ByVal provider As IFormatProvider) As Boolean

Implements Convertible.ToBoolean

[JScript] function Convertible.ToBoolean(provider : IFormatProvider) : Boolean;

Convertible.ToByte

[C#] byte Convertible.ToByte(IFormatProvider provider);

[C++] unsigned char Convertible::ToByte(IFormatProvider* provider);

[VB] Function ToByte(ByVal provider As IFormatProvider) As Byte Implements

1 IConvertible.ToByte

2 [JScript] function IConvertible.ToByte(provider : IFormatProvider) : Byte;

3 IConvertible.ToChar

4
5 [C#] char IConvertible.ToChar(IFormatProvider provider);

6 [C++] __wchar_t IConvertible::ToChar(IFormatProvider* provider);

7 [VB] Function ToChar(ByVal provider As IFormatProvider) As Char Implements

8 IConvertible.ToChar

9 [JScript] function IConvertible.ToChar(provider : IFormatProvider) : Char;

10 IConvertible.ToDateTime

11
12 [C#] DateTime IConvertible.ToDateTime(IFormatProvider provider);

13 [C++] DateTime IConvertible::ToDateTime(IFormatProvider* provider);

14 [VB] Function ToDateTime(ByVal provider As IFormatProvider) As DateTime

15 Implements IConvertible.ToDateTime

16 [JScript] function IConvertible.ToDateTime(provider : IFormatProvider) :

17 DateTime;

18 IConvertible.ToDecimal

19
20 [C#] decimal IConvertible.ToDecimal(IFormatProvider provider);

21 [C++] Decimal IConvertible::ToDecimal(IFormatProvider* provider);

22 [VB] Function ToDecimal(ByVal provider As IFormatProvider) As Decimal

23 Implements IConvertible.ToDecimal

24 [JScript] function IConvertible.ToDecimal(provider : IFormatProvider) : Decimal;

25 IConvertible.ToDouble

```

1
2 [C#] double IConvertible.ToDouble(IFormatProvider provider);
3 [C++] double IConvertible::ToDouble(IFormatProvider* provider);
4 [VB] Function ToDouble(ByVal provider As IFormatProvider) As Double
5 Implements IConvertible.ToDouble
6 [JScript] function IConvertible.ToDouble(provider : IFormatProvider) : double;
7     IConvertible.ToInt16
8
9 [C#] short IConvertible.ToInt16(IFormatProvider provider);
10 [C++] short IConvertible::ToInt16(IFormatProvider* provider);
11 [VB] Function ToInt16(ByVal provider As IFormatProvider) As Short
12 Implements IConvertible.ToInt16
13 [JScript] function IConvertible.ToInt16(provider : IFormatProvider) : Int16;
14     IConvertible.ToInt32
15
16 [C#] int IConvertible.ToInt32(IFormatProvider provider);
17 [C++] int IConvertible::ToInt32(IFormatProvider* provider);
18 [VB] Function ToInt32(ByVal provider As IFormatProvider) As Integer
19 Implements IConvertible.ToInt32
20 [JScript] function IConvertible.ToInt32(provider : IFormatProvider) : int;
21     IConvertible.ToInt64
22
23 [C#] long IConvertible.ToInt64(IFormatProvider provider);
24 [C++] __int64 IConvertible::ToInt64(IFormatProvider* provider);
25 [VB] Function ToInt64(ByVal provider As IFormatProvider) As Long Implements

```

1 IConvertible.ToInt64

2 [JScript] function IConvertible.ToInt64(provider : IFormatProvider) : long;

3 IConvertible.ToSByte

5 [C#] sbyte IConvertible.ToSByte(IFormatProvider provider);

6 [C++] char IConvertible::ToSByte(IFormatProvider* provider);

7 [VB] Function ToSByte(ByVal provider As IFormatProvider) As SByte

8 Implements IConvertible.ToSByte

9 [JScript] function IConvertible.ToSByte(provider : IFormatProvider) : SByte;

10 IConvertible.ToSingle

12 [C#] float IConvertible.ToSingle(IFormatProvider provider);

13 [C++] float IConvertible::ToSingle(IFormatProvider* provider);

14 [VB] Function ToSingle(ByVal provider As IFormatProvider) As Single

15 Implements IConvertible.ToSingle

16 [JScript] function IConvertible.ToSingle(provider : IFormatProvider) : float;

17 IConvertible.ToType

19 [C#] object IConvertible.ToType(Type type, IFormatProvider provider);

20 [C++] Object* IConvertible::ToType(Type* type, IFormatProvider* provider);

21 [VB] Function ToType(ByVal type As Type, ByVal provider As IFormatProvider)

22 As Object Implements IConvertible.ToType

23 [JScript] function IConvertible.ToType(type : Type, provider : IFormatProvider) :

24 Object;

25 IConvertible.ToUInt16

```

1
2 [C#] ushort IConvertible.ToUInt16(IFormatProvider provider);
3 [C++] unsigned short IConvertible::ToUInt16(IFormatProvider* provider);
4 [VB] Function ToUInt16(ByVal provider As IFormatProvider) As UInt16
5 Implements IConvertible.ToUInt16
6 [JScript] function IConvertible.ToUInt16(provider : IFormatProvider) : UInt16;
7     IConvertible.ToUInt32
8
9 [C#] uint IConvertible.ToUInt32(IFormatProvider provider);
10 [C++] unsigned int IConvertible::ToUInt32(IFormatProvider* provider);
11 [VB] Function ToUInt32(ByVal provider As IFormatProvider) As UInt32
12 Implements IConvertible.ToUInt32
13 [JScript] function IConvertible.ToUInt32(provider : IFormatProvider) : UInt32;
14     IConvertible.ToUInt64
15
16 [C#] ulong IConvertible.ToUInt64(IFormatProvider provider);
17 [C++] unsigned __int64 IConvertible::ToUInt64(IFormatProvider* provider);
18 [VB] Function ToUInt64(ByVal provider As IFormatProvider) As UInt64
19 Implements IConvertible.ToUInt64
20 [JScript] function IConvertible.ToUInt64(provider : IFormatProvider) : UInt64;
21     ToString
22
23 [C#] public override string ToString();
24 [C++] public: String* ToString();
25 [VB] Overrides Public Function ToString() As String

```

[JScript] public override function ToString() : String; Converts the numeric value of this instance to its equivalent **String** representation.

Description

Converts the numeric value of this instance to its equivalent **String** representation.

Return Value: The **System.String** representation of the value of this instance.

The return value can be

System.Globalization.NumberFormatInfo.PositiveInfinitySymbol ,
System.Globalization.NumberFormatInfo.NegativeInfinitySymbol ,
System.Globalization.NumberFormatInfo.NaNSymbol , or a string of the form:
[sign]integral-digits[.[fractional-digits]][e[sign]exponential-digits] Optional items are framed in square brackets ('[' and ']'). Items containing the term "digits" consist of a series of numeric characters ranging from 0 to 9.

ToString

[C#] public string ToString(IFormatProvider provider);
[C++] public: __sealed String* ToString(IFormatProvider* provider);
[VB] NotOverridable Public Function ToString(ByVal provider As
IFormatProvider) As String
[JScript] public function ToString(provider : IFormatProvider) : String;

Description

Converts the numeric value of this instance to its equivalent **String** representation using the specified culture-specific format information.

Return Value: The **System.String** representation of the value of this instance as specified by *provider* .

The return value can be

System.Globalization.NumberFormatInfo.PositiveInfinitySymbol ,

System.Globalization.NumberFormatInfo.NegativeInfinitySymbol ,

System.Globalization.NumberFormatInfo.NaNSymbol , or a string of the form:

[sign]integral-digits[.[fractional-digits]][e[sign]exponential-digits] Optional items are framed in square brackets ('[' and ']'). Items containing the term "digits" consist of a series of numeric characters ranging from 0 to 9. An

System.IFormatProvider interface implementation which supplies culture-specific formatting information.

ToString

[C#] public string ToString(string format);

[C++] public: String* ToString(String* format);

[VB] Public Function ToString(ByVal format As String) As String

[JScript] public function ToString(format : String) : String;

Description

Converts the numeric value of this instance to its equivalent **String** representation, using the specified format.

Return Value: The **System.String** representation of the value of this instance as specified by *format* .

The return value can be

System.Globalization.NumberFormatInfo.PositiveInfinitySymbol ,

System.Globalization.NumberFormatInfo.NegativeInfinitySymbol ,
System.Globalization.NumberFormatInfo.NaNSymbol , or a string of the form:
[sign]integral-digits[.[fractional-digits]][e[sign]exponential-digits] Optional items
are framed in square brackets ('[' and ']'). Items containing the term "digits" consist
of a series of numeric characters ranging from 0 to 9. A format string.

ToString

[C#] public string ToString(string format, IFormatProvider provider);

[C++] public: __sealed String* ToString(String* format, IFormatProvider*
provider);

[VB] NotOverridable Public Function ToString(ByVal format As String, ByVal
provider As IFormatProvider) As String

[JScript] public function ToString(format : String, provider : IFormatProvider) :
String;

Description

Converts the numeric value of this instance to its equivalent **String**
representation using the specified format and culture-specific format information.

Return Value: The **System.String** representation of the value of this instance as
specified by *format* and *provider* .

The return value can be

System.Globalization.NumberFormatInfo.PositiveInfinitySymbol ,

System.Globalization.NumberFormatInfo.NegativeInfinitySymbol ,

System.Globalization.NumberFormatInfo.NaNSymbol , or a string of the form:

[sign]integral-digits[.[fractional-digits]][e[sign]exponential-digits] Optional items

are framed in square brackets ('[' and ']'). Items containing the term "digits" consist of a series of numeric characters ranging from 0 to 9. A format specification. An **System.IFormatProvider** interface implementation which supplies culture-specific formatting information.

TryParse

[C#] public static bool TryParse(string s, NumberStyles style, IFormatProvider provider, out double result);

[C++] public: static bool TryParse(String* s, NumberStyles style, IFormatProvider* provider, double* result);

[VB] Public Shared Function TryParse(ByVal s As String, ByVal style As NumberStyles, ByVal provider As IFormatProvider, ByRef result As Double) As Boolean

[JScript] public static function TryParse(s : String, style : NumberStyles, provider : IFormatProvider, result : double) : Boolean;

Description

Converts the **String** representation of a number in a specified style and culture-specific format to its double-precision floating point number equivalent.

Return Value: **true** if *s* is converted successfully; otherwise, **false** .

The **system.Double.TryParse** method is like the **system.Double.Parse** method, except this method does not throw an exception if the conversion fails. If the conversion succeeds, the return value is **true** and the *result* parameter is set to the outcome of the conversion. If the conversion fails, the return value is **false** and the *result* parameter is set to zero. A **System.String** containing a number to

convert. The combination of one or more

System.Globalization.NumberStyles constants that indicate the permitted format of *s*. An **System.IFormatProvider** interface implementation which supplies culture-specific formatting information about *s*. A double-precision floating-point number equivalent to the numeric value or symbol specified in *s*. If the return value is **false**, *result* is set to zero.

DuplicateWaitObjectException class (System)

TryParse

Description

The exception that is thrown when an object appears more than once in an array of synchronization objects.

The common language runtime provides a thread synchronization mechanism based on synchronization objects waiting for execution in an array of **System.Threading.WaitHandle** objects. If the array of **System.Threading.WaitHandle** objects passed to **System.Threading.WaitHandle.WaitAll(System.Threading.WaitHandle[],System.Int32,System.Boolean)** or **System.Threading.WaitHandle.WaitAny(System.Threading.WaitHandle[],System.Int32,System.Boolean)** contains any duplicate operating system handles, **System.DuplicateWaitObjectException** is thrown. For more information, see **System.Threading.WaitHandle**.

DuplicateWaitObjectException

Example Syntax:

TryParse

[C#] public DuplicateWaitObjectException();

[C++] public: DuplicateWaitObjectException();

[VB] Public Sub New()

[JScript] public function DuplicateWaitObjectException(); Initializes a new instance of the **System.DuplicateWaitObjectException** class.

Description

Initializes a new instance of the **System.DuplicateWaitObjectException** class with default properties.

The following table shows the initial property values for an instance of **System.DuplicateWaitObjectException**.

DuplicateWaitObjectException

Example Syntax:

TryParse

[C#] public DuplicateWaitObjectException(string parameterName);

[C++] public: DuplicateWaitObjectException(String* parameterName);

[VB] Public Sub New(ByVal parameterName As String)

[JScript] public function DuplicateWaitObjectException(parameterName : String);

Description

Initializes a new instance of the **System.DuplicateWaitObjectException** class with the name of the parameter that causes this exception.

The following table shows the initial property values for an instance of **System.DuplicateWaitObjectException** . The name of the invalid parameter.

DuplicateWaitObjectException

Example Syntax:

TryParse

[C#] protected DuplicateWaitObjectException(SerializationInfo info, StreamingContext context);

[C++] protected: DuplicateWaitObjectException(SerializationInfo* info, StreamingContext context);

[VB] Protected Sub New(ByVal info As SerializationInfo, ByVal context As StreamingContext)

[JScript] protected function DuplicateWaitObjectException(info : SerializationInfo, context : StreamingContext);

Description

Initializes a new instance of the **System.DuplicateWaitObjectException** class with serialized data.

This constructor is called during deserialization to reconstitute the exception object transmitted over a stream. For more information, see . The object that holds the serialized object data. The contextual information about the source or destination.

DuplicateWaitObjectException

Example Syntax:

TryParse

```

1
2 [C#] public DuplicateWaitObjectException(string parameterName, string
3 message);
4 [C++] public: DuplicateWaitObjectException(String* parameterName, String*
5 message);
6 [VB] Public Sub New(ByVal parameterName As String, ByVal message As
7 String)
8 [JScript] public function DuplicateWaitObjectException(parameterName : String,
9 message : String);
10

```

Description

Initializes a new instance of the **System.DuplicateWaitObjectException** class with a specified error message and the name of the parameter that causes this exception.

When an **Exception** *X* is thrown as a direct result of a previous exception *Y*, the **System.Exception.InnerException** property of *X* should contain a reference to *Y*. The **InnerException** property returns the same value as was passed into the constructor, or **null** if the inner exception value was not supplied to the constructor. The name of the invalid parameter. The error message that explains the reason for the exception.

HelpLink

HResult

InnerException

Message

ParamName

Source

StackTrace

TargetSite

EntryPointNotFoundException class (System)

ToString

Description

The exception that is thrown when an attempt to load a class fails due to the absence of an entry method.

For a list of initial property values for an instance of **System.EntryPointNotFoundException**, see the **System.EntryPointNotFoundException.#ctor** constructors.

EntryPointNotFoundException

Example Syntax:

ToString

[C#] public EntryPointNotFoundException();

[C++] public: EntryPointNotFoundException();

[VB] Public Sub New()

[JScript] public function EntryPointNotFoundException(); Initializes a new instance of the **System.EntryPointNotFoundException** class.

Description

1 Initializes a new instance of the **System.EntryPointNotFoundException**
2 class with default properties.

3 The following table shows the initial property values for an instance of
4 **System.EntryPointNotFoundException** .

5 EntryPointNotFoundException

6 *Example Syntax:*

7 ToString

9 [C#] public EntryPointNotFoundException(string message);

10 [C++] public: EntryPointNotFoundException(String* message);

11 [VB] Public Sub New(ByVal message As String)

12 [JScript] public function EntryPointNotFoundException(message : String);

13
14 *Description*

15 Initializes a new instance of the **System.EntryPointNotFoundException**
16 class with a specified error message.

17 The following table shows the initial property values for an instance of
18 **System.EntryPointNotFoundException** . The error message that explains the
19 reason for the exception.

20 EntryPointNotFoundException

21 *Example Syntax:*

22 ToString

23
24 [C#] protected EntryPointNotFoundException(SerializationInfo info,
25 StreamingContext context);

1 [C++] protected: EntryPointNotFoundException(SerializationInfo* info,
 2 StreamingContext context);
 3 [VB] Protected Sub New(ByVal info As SerializationInfo, ByVal context As
 4 StreamingContext)
 5 [JScript] protected function EntryPointNotFoundException(info :
 6 SerializationInfo, context : StreamingContext);

8 *Description*

9 Initializes a new instance of the **System.EntryPointNotFoundException**
 10 class with serialized data.

11 This constructor is called during deserialization to reconstitute the
 12 exception object transmitted over a stream. For more information, see . The object
 13 that holds the serialized object data. The contextual information about the source
 14 or destination.

15 EntryPointNotFoundException

16 *Example Syntax:*

17 ToString

18
 19 [C#] public EntryPointNotFoundException(string message, Exception inner);
 20 [C++] public: EntryPointNotFoundException(String* message, Exception* inner);
 21 [VB] Public Sub New(ByVal message As String, ByVal inner As Exception)
 22 [JScript] public function EntryPointNotFoundException(message : String, inner :
 23 Exception);

24 25 *Description*

1 Initializes a new instance of the **System.EntryPointNotFoundException**
2 class with a specified error message and a reference to the inner exception that is
3 the root cause of this exception.

4 When an **Exception** *X* is thrown as a direct result of a previous exception *Y* ,
5 the **System.Exception.InnerException** property of *X* should contain a reference
6 to *Y* . The **InnerException** property returns the same value as was passed into the
7 constructor, or **null** if the inner exception value was not supplied to the
8 constructor. The error message that explains the reason for the exception. An
9 instance of **System.Exception** that is the cause of the current **Exception**. If *inner*
10 is non-null, then the current **Exception** is raised in a catch block handling *inner* .

11 HelpLink

12 HResult

13 InnerException

14 Message

15 Source

16 StackTrace

17 TargetSite

18 TypeName

19 Enum class (System)

20 ToString

21
22
23 *Description*

24 Provides the base class for enumerations.

1 An enumeration type is a named constant whose underlying type is any
2 integral type except **System.Char** . Programming languages typically provide
3 syntax to declare an enumeration that consists of a set of named constants and
4 their values.

5 Enum

6 *Example Syntax:*

7 ToString

8
9 [C#] protected Enum();

10 [C++] protected: Enum();

11 [VB] Protected Sub New()

12 [JScript] protected function Enum();

13 CompareTo

14
15 [C#] public int CompareTo(object target);

16 [C++] public: __sealed int CompareTo(Object* target);

17 [VB] NotOverridable Public Function CompareTo(ByVal target As Object) As

18 Integer

19 [JScript] public function CompareTo(target : Object) : int;

20
21 *Description*

22 Compares this instance to a specified object and returns an indication of
23 their relative values.

24 *Return Value:* A signed number indicating the relative values of this instance and
25 *target* . An object to compare, or **null**.

Equals

[C#] public override bool Equals(object obj);

[C++] public: bool Equals(Object* obj);

[VB] Overrides Public Function Equals(ByVal obj As Object) As Boolean

[JScript] public override function Equals(obj : Object) : Boolean;

Description

Returns a value indicating whether this instance is equal to a specified object.

Return Value: **true** if *obj* is an **Enum** with the same underlying type and value as this instance; otherwise, **false** . An object to compare with this instance.

Format

[C#] public static string Format(Type enumType, object value, string format);

[C++] public: static String* Format(Type* enumType, Object* value, String* format);

[VB] Public Shared Function Format(ByVal enumType As Type, ByVal value As Object, ByVal format As String) As String

[JScript] public static function Format(enumType : Type, value : Object, format : String) : String;

Description

Converts the specified value of a specified enumerated type to its equivalent **String** representation according to the specified format.

Return Value: A string representation of *value* .

The valid format values are: Format Description "G" or "g" If *value* is equal to a named enumerated constant, the name of that constant is returned; otherwise, the decimal equivalent of *value* is returned. The enumeration type of the value to be converted. The value to be converted. The output format to use.

GetHashCode

[C#] public override int GetHashCode();

[C++] public: int GetHashCode();

[VB] Overrides Public Function GetHashCode() As Integer

[JScript] public override function GetHashCode() : int;

Description

Returns the hash code for this instance.

Return Value: A 32-bit signed integer hash code.

GetName

[C#] public static string GetName(Type enumType, object value);

[C++] public: static String* GetName(Type* enumType, Object* value);

[VB] Public Shared Function GetName(ByVal enumType As Type, ByVal value As Object) As String

[JScript] public static function GetName(enumType : Type, value : Object) :

String;

Description

Retrieves the name of the constant in the specified enumeration that has the specified value.

Return Value: A **System.String** containing the name of the enumerated constant in *enumType* whose value is *value* , or **null** if no such constant is found. An enumeration. The value of a particular enumerated constant in terms of its underlying type.

GetNames

```
[C#] public static string[] GetNames(Type enumType);
```

```
[C++] public: static String* GetNames(Type* enumType) __gc[];
```

```
[VB] Public Shared Function GetNames(ByVal enumType As Type) As String()
```

```
[JScript] public static function GetNames(enumType : Type) : String[];
```

Description

Retrieves an array of the names of the constants in a specified enumeration.

Return Value: A **System.String** array of the names of the constants in *enumType* . The elements of the array are sorted by the values of the enumerated constants. An enumeration.

GetTypeCode

```
[C#] public TypeCode GetTypeCode();
```

```
[C++] public: __sealed TypeCode GetTypeCode();
```

```
[VB] NotOverridable Public Function GetTypeCode() As TypeCode
```

1 [JScript] public function GetTypeCode() : TypeCode;

3 *Description*

4 Returns the underlying **TypeCode** for this instance.

5 *Return Value:* The **System.TypeCode** for this instance.

6 GetUnderlyingType

8 [C#] public static Type GetUnderlyingType(Type enumType);

9 [C++] public: static Type* GetUnderlyingType(Type* enumType);

10 [VB] Public Shared Function GetUnderlyingType(ByVal enumType As Type) As
11 Type

12 [JScript] public static function GetUnderlyingType(enumType : Type) : Type;

14 *Description*

15 Returns the underlying type of the specified enumeration.

16 *Return Value:* The underlying **System.Type** of *enumType* . An enumerated type.

17 GetValues

19 [C#] public static Array GetValues(Type enumType);

20 [C++] public: static Array* GetValues(Type* enumType);

21 [VB] Public Shared Function GetValues(ByVal enumType As Type) As Array

22 [JScript] public static function GetValues(enumType : Type) : Array;

24 *Description*

Retrieves an array of the values of the constants in a specified enumeration.

Return Value: An **Array** of the values of the constants in *enumType* . The elements of the array are sorted by the values of the enumeration constants. An enumeration.

IsDefined

[C#] public static bool IsDefined(Type enumType, object value);

[C++] public: static bool IsDefined(Type* enumType, Object* value);

[VB] Public Shared Function IsDefined(ByVal enumType As Type, ByVal value As Object) As Boolean

[JScript] public static function IsDefined(enumType : Type, value : Object) : Boolean;

Description

Returns an indication whether a constant with a specified value exists in a specified enumeration.

Return Value: **true** if a constant in *enumType* has a value equal to *value* ; otherwise, **false** . An enumeration. The value or name of a constant in *enumType* .

Parse

[C#] public static object Parse(Type enumType, string value);

[C++] public: static Object* Parse(Type* enumType, String* value);

[VB] Public Shared Function Parse(ByVal enumType As Type, ByVal value As String) As Object

[JScript] public static function Parse(enumType : Type, value : String) : Object;

Converts the **String** representation of the name or numeric value of one or more enumerated constants to an equivalent enumerated object.

Description

Converts the **String** representation of the name or numeric value of one or more enumerated constants to an equivalent enumerated object.

Return Value: An object of type *enumType* whose value is represented by *value* .

value contains a value, a named constant, or a list of named constants delimited by commas (","). One or more blanks (" ") can precede or follow each value, name, or comma in *value* . If *value* is a list, the return value is the value of the specified names combined with a bitwise OR operation. The **System.Type** of the enumeration. A **System.String** containing the name or value to convert.

Parse

[C#] public static object Parse(Type enumType, string value, bool ignoreCase);
[C++] public: static Object* Parse(Type* enumType, String* value, bool ignoreCase);
[VB] Public Shared Function Parse(ByVal enumType As Type, ByVal value As String, ByVal ignoreCase As Boolean) As Object
[JScript] public static function Parse(enumType : Type, value : String, ignoreCase : Boolean) : Object;

Description

Converts the **String** representation of the name or numeric value of one or more enumerated constants to an equivalent enumerated object. A parameter

1 specifies whether the operation is case-sensitive.

2 *Return Value:* An object of type *enumType* whose value is represented by *value* .

3 *value* contains a value, a named constant, or a list of named constants
4 delimited by commas (","). One or more blanks (" ") can precede or follow each
5 value, name, or comma in *value* . If *value* is a list, the return value is the value of
6 the specified names combined with a bitwise OR operation. The **System.Type** of
7 the enumeration. A **System.String** containing the name or value to convert. If
8 **true**, ignore case; otherwise, regard case.

9 **Convertible.ToBoolean**

10
11 [C#] bool Convertible.ToBoolean(IFormatProvider provider);

12 [C++] bool Convertible::ToBoolean(IFormatProvider* provider);

13 [VB] Function ToBoolean(ByVal provider As IFormatProvider) As Boolean

14 Implements Convertible.ToBoolean

15 [JScript] function Convertible.ToBoolean(provider : IFormatProvider) : Boolean;

16 **Convertible.ToByte**

17
18 [C#] byte Convertible.ToByte(IFormatProvider provider);

19 [C++] unsigned char Convertible::ToByte(IFormatProvider* provider);

20 [VB] Function ToByte(ByVal provider As IFormatProvider) As Byte Implements

21 Convertible.ToByte

22 [JScript] function Convertible.ToByte(provider : IFormatProvider) : Byte;

23 **Convertible.ToChar**

24
25 [C#] char Convertible.ToChar(IFormatProvider provider);

```

1 [C++] __wchar_t IConvertible::ToChar(IFormatProvider* provider);
2 [VB] Function ToChar(ByVal provider As IFormatProvider) As Char Implements
3 IConvertible.ToChar
4 [JScript] function IConvertible.ToChar(provider : IFormatProvider) : Char;
5     IConvertible.ToDateTime
6
7 [C#] DateTime IConvertible.ToDateTime(IFormatProvider provider);
8 [C++] DateTime IConvertible::ToDateTime(IFormatProvider* provider);
9 [VB] Function ToDateTime(ByVal provider As IFormatProvider) As DateTime
10 Implements IConvertible.ToDateTime
11 [JScript] function IConvertible.ToDateTime(provider : IFormatProvider) :
12 DateTime;
13     IConvertible.ToDecimal
14
15 [C#] decimal IConvertible.ToDecimal(IFormatProvider provider);
16 [C++] Decimal IConvertible::ToDecimal(IFormatProvider* provider);
17 [VB] Function ToDecimal(ByVal provider As IFormatProvider) As Decimal
18 Implements IConvertible.ToDecimal
19 [JScript] function IConvertible.ToDecimal(provider : IFormatProvider) : Decimal;
20     IConvertible.ToDouble
21
22 [C#] double IConvertible.ToDouble(IFormatProvider provider);
23 [C++] double IConvertible::ToDouble(IFormatProvider* provider);
24 [VB] Function ToDouble(ByVal provider As IFormatProvider) As Double
25

```

1 Implements IConvertible.ToDouble

2 [JScript] function IConvertible.ToDouble(provider : IFormatProvider) : double;

3 IConvertible.ToInt16

5 [C#] short IConvertible.ToInt16(IFormatProvider provider);

6 [C++] short IConvertible::ToInt16(IFormatProvider* provider);

7 [VB] Function ToInt16(ByVal provider As IFormatProvider) As Short

8 Implements IConvertible.ToInt16

9 [JScript] function IConvertible.ToInt16(provider : IFormatProvider) : Int16;

10 IConvertible.ToInt32

12 [C#] int IConvertible.ToInt32(IFormatProvider provider);

13 [C++] int IConvertible::ToInt32(IFormatProvider* provider);

14 [VB] Function ToInt32(ByVal provider As IFormatProvider) As Integer

15 Implements IConvertible.ToInt32

16 [JScript] function IConvertible.ToInt32(provider : IFormatProvider) : int;

17 IConvertible.ToInt64

19 [C#] long IConvertible.ToInt64(IFormatProvider provider);

20 [C++] __int64 IConvertible::ToInt64(IFormatProvider* provider);

21 [VB] Function ToInt64(ByVal provider As IFormatProvider) As Long Implements

22 IConvertible.ToInt64

23 [JScript] function IConvertible.ToInt64(provider : IFormatProvider) : long;

24 IConvertible.ToSByte

```

1
2 [C#] sbyte IConvertible.ToSByte(IFormatProvider provider);
3 [C++] char IConvertible::ToSByte(IFormatProvider* provider);
4 [VB] Function ToSByte(ByVal provider As IFormatProvider) As SByte
5 Implements IConvertible.ToSByte
6 [JScript] function IConvertible.ToSByte(provider : IFormatProvider) : SByte;
7     IConvertible.ToSingle
8
9 [C#] float IConvertible.ToSingle(IFormatProvider provider);
10 [C++] float IConvertible::ToSingle(IFormatProvider* provider);
11 [VB] Function ToSingle(ByVal provider As IFormatProvider) As Single
12 Implements IConvertible.ToSingle
13 [JScript] function IConvertible.ToSingle(provider : IFormatProvider) : float;
14     IConvertible.ToType
15
16 [C#] object IConvertible.ToType(Type type, IFormatProvider provider);
17 [C++] Object* IConvertible::ToType(Type* type, IFormatProvider* provider);
18 [VB] Function ToType(ByVal type As Type, ByVal provider As IFormatProvider)
19 As Object Implements IConvertible.ToType
20 [JScript] function IConvertible.ToType(type : Type, provider : IFormatProvider) :
21 Object;
22     IConvertible.ToUInt16
23
24 [C#] ushort IConvertible.ToUInt16(IFormatProvider provider);
25 [C++] unsigned short IConvertible::ToUInt16(IFormatProvider* provider);
    
```

```

1  [VB] Function ToUInt16(ByVal provider As IFormatProvider) As UInt16
2  Implements IConvertible.ToUInt16
3  [JScript] function IConvertible.ToUInt16(provider : IFormatProvider) : UInt16;
4      IConvertible.ToUInt32
5
6  [C#] uint IConvertible.ToUInt32(IFormatProvider provider);
7  [C++] unsigned int IConvertible::ToUInt32(IFormatProvider* provider);
8  [VB] Function ToUInt32(ByVal provider As IFormatProvider) As UInt32
9  Implements IConvertible.ToUInt32
10 [JScript] function IConvertible.ToUInt32(provider : IFormatProvider) : UInt32;
11     IConvertible.ToUInt64
12
13 [C#] ulong IConvertible.ToUInt64(IFormatProvider provider);
14 [C++] unsigned __int64 IConvertible::ToUInt64(IFormatProvider* provider);
15 [VB] Function ToUInt64(ByVal provider As IFormatProvider) As UInt64
16 Implements IConvertible.ToUInt64
17 [JScript] function IConvertible.ToUInt64(provider : IFormatProvider) : UInt64;
18     ToObject
19
20 [C#] public static object ToObject(Type enumType, byte value);
21 [C++] public: static Object* ToObject(Type* enumType, unsigned char value);
22 [VB] Public Shared Function ToObject(ByVal enumType As Type, ByVal value
23     As Byte) As Object
24 [JScript] public static function ToObject(enumType : Type, value : Byte) : Object;
25

```

Description

Returns an instance of the specified enumeration type set to the specified 8-bit unsigned integer value.

Return Value: An instance of the enumeration set to *value* . The enumeration for which to create a value. The value to set.

ToObject

[C#] public static object ToObject(Type enumType, short value);

[C++] public: static Object* ToObject(Type* enumType, short value);

[VB] Public Shared Function ToObject(ByVal enumType As Type, ByVal value As Short) As Object

[JScript] public static function ToObject(enumType : Type, value : Int16) : Object;

Description

Returns an instance of the specified enumeration type set to the specified 16-bit signed integer value.

Return Value: An instance of the enumeration set to *value* . The enumeration for which to create a value. The value to set.

ToObject

[C#] public static object ToObject(Type enumType, int value);

[C++] public: static Object* ToObject(Type* enumType, int value);

[VB] Public Shared Function ToObject(ByVal enumType As Type, ByVal value As Integer) As Object

1 [JScript] public static function ToObject(enumType : Type, value : int) : Object;

3 *Description*

4 Returns an instance of the specified enumeration type set to the specified
5 32-bit signed integer value.

6 *Return Value:* An instance of the enumeration set to *value* . The enumeration for
7 which to create a value. The value to set.

8 ToObject

10 [C#] public static object ToObject(Type enumType, long value);

11 [C++] public: static Object* ToObject(Type* enumType, __int64 value);

12 [VB] Public Shared Function ToObject(ByVal enumType As Type, ByVal value
13 As Long) As Object

14 [JScript] public static function ToObject(enumType : Type, value : long) : Object;

16 *Description*

17 Returns an instance of the specified enumeration type set to the specified
18 64-bit signed integer value.

19 *Return Value:* An instance of the enumeration set to *value* . The enumeration for
20 which to create a value. The value to set.

21 ToObject

23 [C#] public static object ToObject(Type enumType, object value);

24 [C++] public: static Object* ToObject(Type* enumType, Object* value);

25 [VB] Public Shared Function ToObject(ByVal enumType As Type, ByVal value

As Object) As Object

[JScript] public static function ToObject(enumType : Type, value : Object) :

Object; Returns an instance of the specified enumeration set to the specified value.

Description

Returns an instance of the specified enumeration set to the specified value.

Return Value: An enumeration object whose value is *value* .

value is specified in terms of the underlying type of the enumeration. An enumeration. The value.

ToObject

[C#] public static object ToObject(Type enumType, sbyte value);

[C++] public: static Object* ToObject(Type* enumType, char value);

[VB] Public Shared Function ToObject(ByVal enumType As Type, ByVal value As SByte) As Object

[JScript] public static function ToObject(enumType : Type, value : SByte) :

Object; Returns an instance of the specified enumeration type set to the specified value.

Description

Returns an instance of the specified enumeration type set to the specified 8-bit signed integer value.

Return Value: An instance of the enumeration set to *value* . The enumeration for which to create a value. The value to set.

ToObject

[C#] public static object ToObject(Type enumType, ushort value);

[C++] public: static Object* ToObject(Type* enumType, unsigned short value);

[VB] Public Shared Function ToObject(ByVal enumType As Type, ByVal value

As UInt16) As Object

[JScript] public static function ToObject(enumType : Type, value : UInt16) :

Object;

Description

Returns an instance of the specified enumeration type set to the specified 16-bit unsigned integer value.

Return Value: An instance of the enumeration set to *value* . The enumeration for which to create a value. The value to set.

ToObject

[C#] public static object ToObject(Type enumType, uint value);

[C++] public: static Object* ToObject(Type* enumType, unsigned int value);

[VB] Public Shared Function ToObject(ByVal enumType As Type, ByVal value

As UInt32) As Object

[JScript] public static function ToObject(enumType : Type, value : UInt32) :

Object;

Description

Returns an instance of the specified enumeration type set to the specified 32-bit unsigned integer value.

Return Value: An instance of the enumeration set to *value* . The enumeration for which to create a value. The value to set.

ToObject

[C#] public static object ToObject(Type enumType, ulong value);

[C++] public: static Object* ToObject(Type* enumType, unsigned __int64 value);

[VB] Public Shared Function ToObject(ByVal enumType As Type, ByVal value As UInt64) As Object

[JScript] public static function ToObject(enumType : Type, value : UInt64) : Object;

Description

Returns an instance of the specified enumeration type set to the specified 64-bit unsigned integer value.

Return Value: An instance of the enumeration set to *value* . The enumeration for which to create a value. The value to set.

ToString

[C#] public override string ToString();

[C++] public: String* ToString();

[VB] Overrides Public Function ToString() As String

[JScript] public override function ToString() : String;

Description

Converts the value of this instance to its equivalent **String** representation.

Return Value: The **String** representation of the value of this instance.

This method behaves as if the general format character, "G", were specified. That is, if the **System.FlagsAttribute** is not applied to this enumerated type and there is a named constant equal to the value of this instance, then the return value is a string containing the name of the constant. If the **System.FlagsAttribute** is applied and there is a combination of one or more named constants equal to the value of this instance, then the return value is a string containing a delimiter-separated list of the names of the constants. Otherwise, the return value is the string representation of the numeric value of this instance.

ToString

[C#] public string ToString(IFormatProvider provider);

[C++] public: __sealed String* ToString(IFormatProvider* provider);

[VB] NotOverridable Public Function ToString(ByVal provider As IFormatProvider) As String

[JScript] public function ToString(provider : IFormatProvider) : String;

Description

Converts the value of this instance to its equivalent **String** representation using the specified format information.

Return Value: The **String** representation of the name of the value of this instance as specified by *provider* .

provider is reserved; it does not participate in this operation and can be specified as **null** . Therefore, this method is equivalent to the

1 **System.Enum.ToString(System.String)** method that takes no parameters.

2 (Reserved) An **System.IFormatProvider** object that supplies format information
3 about this instance.

4 ToString

5
6 [C#] public string ToString(string format);

7 [C++] public: String* ToString(String* format);

8 [VB] Public Function ToString(ByVal format As String) As String

9 [JScript] public function ToString(format : String) : String; Converts the value of
10 this instance to its equivalent **String** representation.

11
12 *Description*

13 Converts the value of this instance to its equivalent **String** representation
14 using the specified format.

15 *Return Value:* The **String** representation of the value of this instance as specified
16 by *format* .

17 *format* can contain format characters "G" or "g", "D" or "d", "X" or "x",
18 and "F" or "f". If *format* is **null** or an empty string, the general format specifier
19 ("G") is used. For more information about these format characters, see the
20 Remarks section of the

21 **System.Enum.Format(System.Type, System.Object, System.String)** method.

22 For more information about formatting in general, see . A format string.

23 ToString

24
25 [C#] public string ToString(string format, IFormatProvider provider);

[C++] public: __sealed String* ToString(String* format, IFormatProvider* provider);

[VB] NotOverridable Public Function ToString(ByVal format As String, ByVal provider As IFormatProvider) As String

[JScript] public function ToString(format : String, provider : IFormatProvider) : String;

Description

Converts the value of this instance to its equivalent **String** representation using the specified format and format information.

Return Value: The **String** representation of the value of this instance as specified by *format* and *provider* .

format can contain format characters "G" or "g", "D" or "d", "X" or "x", and "F" or "f". If *format* is **null** or an empty string, the general format specifier ("G") is used. For more information about these format characters, see the Remarks section of the **System.Enum.Format(System.Type,System.Object,System.String)** method. For more information about formatting in general, see . A format specification. (Reserved) An **System.IFormatProvider** object that supplies format information about this instance.

Environment class (System)

ToString

Description

Provides information about, and means to manipulate, the current environment and platform.

CommandLine

ToString

[C#] public static string CommandLine {get;}

[C++] public: __property static String* get_CommandLine();

[VB] Public Shared ReadOnly Property CommandLine As String

[JScript] public static function get CommandLine() : String;

Description

Gets the command line for this process.

CurrentDirectory

ToString

[C#] public static string CurrentDirectory {get; set;}

[C++] public: __property static String* get_CurrentDirectory();public: __property static void set_CurrentDirectory(String*);

[VB] Public Shared Property CurrentDirectory As String

[JScript] public static function get CurrentDirectory() : String;public static function set CurrentDirectory(String);

Description

Gets and sets the fully qualified path of the current directory; that is, the directory from which this process starts.

By definition, if this process starts in the root directory of a local or network drive, the value of this property is the drive name followed by a trailing slash (for example, "C:\"). If this process starts in a subdirectory, the value of this property is the drive and subdirectory path, without a trailing slash (for example, "C:\mySubDirectory").

ExitCode

ToString

[C#] public static int ExitCode {get; set;}

[C++] public: __property static int get_ExitCode();public: __property static void set_ExitCode(int);

[VB] Public Shared Property ExitCode As Integer

[JScript] public static function get ExitCode() : int;public static function set ExitCode(int);

Description

Gets or sets the exit code of the process.

This property can be used to return a success code from an application. For example, it can be used to control the execution of a set of applications invoked in a script.

HasShutdownStarted

ToString

[C#] public bool HasShutdownStarted {get;}

[C++] public: __property bool get_HasShutdownStarted();

1 [VB] Public ReadOnly Property HasShutdownStarted As Boolean

2 [JScript] public function get HasShutdownStarted() : Boolean;

3
4 *Description*

5
6 MachineName

7 ToString

8
9 [C#] public static string MachineName {get;}

10 [C++] public: __property static String* get_MachineName();

11 [VB] Public Shared ReadOnly Property MachineName As String

12 [JScript] public static function get MachineName() : String;

13
14 *Description*

15 Gets the NetBIOS name of this local computer.

16 The name of this computer is established at system startup when the name
17 is read from the registry. If this computer is a node in a cluster, the name of the
18 node is returned.

19 NewLine

20 ToString

21
22 [C#] public static string NewLine {get;}

23 [C++] public: __property static String* get_NewLine();

24 [VB] Public Shared ReadOnly Property NewLine As String

25 [JScript] public static function get NewLine() : String;

1
2 *Description*

3 Gets the newline string defined for this environment.

4 The property value is a constant customized specifically for the current
5 platform.

6 OSVersion

7 ToString

8
9 [C#] public static OperatingSystem OSVersion {get;}

10 [C++] public: __property static OperatingSystem* get_OSVersion();

11 [VB] Public Shared ReadOnly Property OSVersion As OperatingSystem

12 [JScript] public static function get OSVersion() : OperatingSystem;

13
14 *Description*

15 Gets an **OperatingSystem** object that contains the current platform
16 identifier and version number.

17 StackTrace

18 ToString

19
20 [C#] public static string StackTrace {get;}

21 [C++] public: __property static String* get_StackTrace();

22 [VB] Public Shared ReadOnly Property StackTrace As String

23 [JScript] public static function get StackTrace() : String;

24
25 *Description*

Gets current stack trace information.

SystemDirectory

ToString

[C#] public static string SystemDirectory {get;}

[C++] public: __property static String* get_SystemDirectory();

[VB] Public Shared ReadOnly Property SystemDirectory As String

[JScript] public static function get SystemDirectory() : String;

Description

Gets the fully qualified path of the system directory.

An example of the value returned is the string "C:\WinNT\System32".

TickCount

ToString

[C#] public static int TickCount {get;}

[C++] public: __property static int get_TickCount();

[VB] Public Shared ReadOnly Property TickCount As Integer

[JScript] public static function get TickCount() : int;

Description

Gets the number of milliseconds elapsed since the system started.

The value of this property is derived from the system timer and is stored as a 32-bit signed integer. Therefore, the elapsed time will wrap around to zero if the system is run continuously for 49.7 days.

UserDomainName

ToString

[C#] public static string UserDomainName {get;}

[C++] public: __property static String* get_UserDomainName();

[VB] Public Shared ReadOnly Property UserDomainName As String

[JScript] public static function get UserDomainName() : String;

Description

Gets the name of the application domain of the current user.

The value of this property is typically the host machine name, but can depend upon the application solution being deployed.

UserInteractive

ToString

[C#] public static bool UserInteractive {get;}

[C++] public: __property static bool get_UserInteractive();

[VB] Public Shared ReadOnly Property UserInteractive As Boolean

[JScript] public static function get UserInteractive() : Boolean;

Description

Gets a value indicating whether the current process is running in user interactive mode.

This will be **false** only when running as a Service Process or from inside a Web application. When this property is **false** , you should not display any modal

dialogs or message boxes, because there is no graphical user interface for the user to interact with.

UserName

ToString

[C#] public static string UserName {get;}

[C++] public: __property static String* get_UserName();

[VB] Public Shared ReadOnly Property UserName As String

[JScript] public static function get UserName() : String;

Description

Gets the user name of the person who started the current thread.

This property can be used to identify the current user to the system and application for security or access purposes. It can also be used to customize a particular application for each user.

Version

ToString

[C#] public static Version Version {get;}

[C++] public: __property static Version* get_Version();

[VB] Public Shared ReadOnly Property Version As Version

[JScript] public static function get Version() : Version;

Description

1 Gets a **Version** object that describes the major, minor, build, and revision
2 numbers of the common language runtime.

3 WorkingSet

4 ToString

5
6 [C#] public static long WorkingSet {get;}

7 [C++] public: __property static __int64 get_WorkingSet();

8 [VB] Public Shared ReadOnly Property WorkingSet As Long

9 [JScript] public static function get WorkingSet() : long;

10
11 *Description*

12 Gets the amount of physical memory mapped to the process context.

13 Exit

14
15 [C#] public static void Exit(int exitCode);

16 [C++] public: static void Exit(int exitCode);

17 [VB] Public Shared Sub Exit(ByVal exitCode As Integer)

18 [JScript] public static function Exit(exitCode : int);

19
20 *Description*

21 Terminates this process and gives the underlying operating system the
22 specified exit code. Exit code to be given to the operating system.

23 ExpandEnvironmentVariables

24
25 [C#] public static string ExpandEnvironmentVariables(string name);

```

1 [C++] public: static String* ExpandEnvironmentVariables(String* name);
2 [VB] Public Shared Function ExpandEnvironmentVariables(ByVal name As
3 String) As String
4 [JScript] public static function ExpandEnvironmentVariables(name : String) :
5 String;
6

```

Description

Replaces the name of each environment variable embedded in the specified string with the string equivalent of the value of the variable, then returns the resulting string.

Return Value: A **System.String** with each environment variable replaced by its value.

Replacement only occurs for environment variables that are set. For example, suppose *name* is "MyENV = %MyENV%". If the environment variable, MyENV, is set to 42, this method returns "MyENV = 42". If MyENV is not set, no change occurs; this method returns "MyENV = %MyENV%". A string containing the names of zero or more environment variables. Each environment variable is quoted with the percent sign character ('%').

GetCommandLineArgs

```

21 [C#] public static string[] GetCommandLineArgs();
22 [C++] public: static String* GetCommandLineArgs() __gc[];
23 [VB] Public Shared Function GetCommandLineArgs() As String()
24 [JScript] public static function GetCommandLineArgs() : String[];
25

```


Description

Returns a **String** array containing the command line arguments for the current process.

Return Value: An array of **System.String** where each element contains a command line argument. The first element is the executable file name, and the following zero or more elements contain the remaining command line arguments.

GetEnvironmentVariable

[C#] public static string GetEnvironmentVariable(string variable);

[C++] public: static String* GetEnvironmentVariable(String* variable);

[VB] Public Shared Function GetEnvironmentVariable(ByVal variable As String)

As String

[JScript] public static function GetEnvironmentVariable(variable : String) : String;

Description

Returns the value of the specified environment variable.

Return Value: A string containing the value of *variable* , or **null** if *variable* is not found. A string containing the name of an environment variable.

GetEnvironmentVariables

[C#] public static IDictionary GetEnvironmentVariables();

[C++] public: static IDictionary* GetEnvironmentVariables();

[VB] Public Shared Function GetEnvironmentVariables() As IDictionary

[JScript] public static function GetEnvironmentVariables() : IDictionary;

Description

Returns all environment variables and their values.

Return Value: An object derived from **System.Collections.IDictionary** , which can return all environment variables and their values.

GetFolderPath

[C#] public static string GetFolderPath(Environment.SpecialFolder folder);

[C++] public: static String* GetFolderPath(Environment.SpecialFolder folder);

[VB] Public Shared Function GetFolderPath(ByVal folder As Environment.SpecialFolder) As String

[JScript] public static function GetFolderPath(folder : Environment.SpecialFolder) : String;

Description

Gets the path to the system special folder identified by the specified enumeration.

This method retrieves the path to a system special folder, such as Program Files, Programs, System, or Startup, which can be used to access common information. The *folder* enumeration specifies the folder to retrieve. Special folders are set by default by the system, or explicitly by the user, when installing a version of Windows. An enumerated constant that identifies a system special folder.

GetLogicalDrives

```

1 [C#] public static string[] GetLogicalDrives();
2
3 [C++] public: static String* GetLogicalDrives() __gc[];
4
5 [VB] Public Shared Function GetLogicalDrives() As String()
6
7 [JScript] public static function GetLogicalDrives() : String[];
8

```

Description

Returns an array of **String** containing the names of the logical drives on the current computer.

Return Value: An array of **System.String** where each element contains the name of a logical drive. For example, if the computer's hard drive is the first logical drive, the first element returned is "C:\".

EventArgs class (System)

ToString

Description

System.EventArgs is the base class for event data.

For more information about events, see the .

ToString

```

22 [C#] public static readonly EventArgs Empty;
23
24 [C++] public: static EventArgs* Empty;
25
26 [VB] Public Shared ReadOnly Empty As EventArgs
27
28 [JScript] public static var Empty : EventArgs;
29

```

1
2 *Description*

3 Represents an event with no event data.

4 The value of **System.EventArgs.Empty** is a read-only instance of

5 **System.EventArgs** .

6 EventArgs

7 *Example Syntax:*

8 ToString

9
10 [C#] public EventArgs();

11 [C++] public: EventArgs();

12 [VB] Public Sub New()

13 [JScript] public function EventArgs();

14
15 *Description*

16 Initializes a new instance of the **System.EventArgs** class.

17 This constructor is only called by the common language runtime.

18 EventHandler delegate (System)

19 ToString

20
21
22 *Description*

23 Represents the method that will handle the event that has no event data. The
24 source of the event. An **EventArgs** that contains the event data.

The event model in the .NET Framework is based on having an event delegate that connects an event with its handler. To raise an event, two elements are needed: A class that holds the event data. This class must derive from the base class **System.EventArgs**.

Exception class (System)

ToString

Description

Defines the base class for all exceptions.

Exceptions are responses to abnormal or exceptional conditions that arise while a program is executing. The common language runtime provides an exception handling model that is based on the representation of exceptions as objects, and the separation of program code and exception handling code into try block and catch block, respectively. There can be one or more catch blocks, each designed to handle a particular type of exception, or one block designed to catch a more specific exception than another block.

Exception

Example Syntax:

ToString

[C#] public Exception();

[C++] public: Exception();

[VB] Public Sub New()

[JScript] public function Exception(); Initializes a new instance of the

System.Exception class.

Description

Initializes a new instance of the **System.Exception** class with default properties.

All the derived classes should provide this default constructor. The following table shows the initial property values for an instance of

System.Exception .

Exception

Example Syntax:

ToString

[C#] public Exception(string message);

[C++] public: Exception(String* message);

[VB] Public Sub New(ByVal message As String)

[JScript] public function Exception(message : String);

Description

Initializes a new instance of the **System.Exception** class with a specified error message.

The following table shows the initial property values for an instance of **System.Exception** . The error message that explains the reason for the exception.

Exception

Example Syntax:

ToString

1
2 [C#] protected Exception(SerializationInfo info, StreamingContext context);

3 [C++] protected: Exception(SerializationInfo* info, StreamingContext context);

4 [VB] Protected Sub New(ByVal info As SerializationInfo, ByVal context As
5 StreamingContext)

6 [JScript] protected function Exception(info : SerializationInfo, context :
7 StreamingContext);

8
9 *Description*

10 Initializes a new instance of the **System.Exception** class with serialized
11 data.

12 This constructor is called during deserialization to reconstitute the
13 exception object transmitted over a stream. For more information, see . The object
14 that holds the serialized object data. The contextual information about the source
15 or destination.

16 Exception

17 *Example Syntax:*

18 ToString

19
20 [C#] public Exception(string message, Exception innerException);

21 [C++] public: Exception(String* message, Exception* innerException);

22 [VB] Public Sub New(ByVal message As String, ByVal innerException As
23 Exception)

24 [JScript] public function Exception(message : String, innerException : Exception);

Description

Initializes a new instance of the **System.Exception** class with a specified error message and a reference to the inner exception that is the root cause of this exception.

When an **Exception** *X* is thrown as a direct result of a previous exception *Y*, the **System.Exception.InnerException** property of *X* should contain a reference to *Y*. The **InnerException** property returns the same value as was passed into the constructor, or **null** if the inner exception value was not supplied to the constructor. The error message that explains the reason for the exception. An instance of **System.Exception** that is the cause of the current **Exception**. If *innerException* is non-null, then the current **Exception** is raised in a catch block handling *innerException*.

HelpLink

ToString

[C#] public virtual string HelpLink {get; set;}

[C++] public: __property virtual String* get_HelpLink();public: __property virtual void set_HelpLink(String*);

[VB] Overridable Public Property HelpLink As String

[JScript] public function get HelpLink() : String;public function set HelpLink(String);

Description

Gets or sets a link to the help file associated with this exception.

The return value, which represents a help file, is a URN or URL. For example: "file:///C:/Applications/Bazzal/help.html#ErrorNum42"

HResult

ToString

[C#] protected int HResult {get; set;}

[C++] protected: __property int get_HResult();protected: __property void set_HResult(int);

[VB] Protected Property HResult As Integer

[JScript] protected function get HResult() : int;protected function set HResult(int);

Description

Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception.

HRESULT is a 32-bit value, divided into three different fields: a severity code, a facility code, and an error code. The severity code indicates whether the return value represents information, warning, or error. The facility code identifies the area of the system responsible for the error. The error code is a unique number that is assigned to represent the exception. Each exception is mapped to a distinct HRESULT. When managed code throws an exception, the runtime passes the HRESULT to the COM client. When unmanaged code returns an error, the HRESULT is converted to an exception, which is then thrown by the runtime.

InnerException

ToString

```

1
2 [C#] public Exception InnerException {get;}
3 [C++] public: __property Exception* get_InnerException();
4 [VB] Public ReadOnly Property InnerException As Exception
5 [JScript] public function get InnerException() : Exception;
6

```

Description

Gets a reference to the inner exception.

You can create a new exception that catches an earlier exception. The code that handles the second exception can make use of the additional information from the earlier exception to handle the error more appropriately.

Message

ToString

```

12
13
14
15 [C#] public virtual string Message {get;}
16 [C++] public: __property virtual String* get_Message();
17 [VB] Overridable Public ReadOnly Property Message As String
18 [JScript] public function get Message() : String;
19

```

Description

Gets the error message text.

Every exception should carry an error message that provides information about the exception, such as why it is thrown. The **System.Exception.Message** property is set when the exception object is constructed. If an exception is

constructed without a supplied error message, this property provides a default message indicating the type of the exception that is thrown.

Source

ToString

[C#] public virtual string Source {get; set;}

[C++] public: __property virtual String* get_Source();public: __property virtual void set_Source(String*);

[VB] Overridable Public Property Source As String

[JScript] public function get Source() : String;public function set Source(String);

Description

Gets or sets a string containing the name of the application or the object that causes the error.

If **System.Exception.Source** is not set, the name of the assembly where the exception originated is returned.

StackTrace

ToString

[C#] public virtual string StackTrace {get;}

[C++] public: __property virtual String* get_StackTrace();

[VB] Overridable Public ReadOnly Property StackTrace As String

[JScript] public function get StackTrace() : String;

Description

1 Gets the stack trace, which identifies the location in the code where the
2 error occurs.

3 The execution stack keeps track of all the methods that are in execution at a
4 given instant. A trace of the method calls is called a stack trace with the most
5 recent method call appearing first. The stack trace listing provides a means to
6 follow the call sequence to the line number in the method where the exception
7 occurs.

8 TargetSite

9 ToString

10
11 [C#] public MethodBase TargetSite {get;}

12 [C++] public: __property MethodBase* get_TargetSite();

13 [VB] Public ReadOnly Property TargetSite As MethodBase

14 [JScript] public function get TargetSite() : MethodBase;

15
16 *Description*

17 Gets the method that throws this exception.

18 If the method that throws this exception is not available and the stack trace
19 is not **null** , **System.Exception.TargetSite** obtains the method from the stack
20 trace. If the stack trace is **null** , **System.Exception.TargetSite** returns **null** .

21 GetBaseException

22
23 [C#] public virtual Exception GetBaseException();

24 [C++] public: virtual Exception* GetBaseException();

25 [VB] Overridable Public Function GetBaseException() As Exception

1 [JScript] public function GetBaseException() : Exception;

3 *Description*

4 Gets the original exception that is thrown.

5 *Return Value:* A reference to the original exception object.

6 **System.Exception.GetBaseException** returns the original, innermost
7 exception that causes this exception and other related exceptions linked via the
8 **System.Exception.InnerException** property. If the current exception is the only
9 one thrown, then its reference will be returned.

10 **GetObjectData**

12 [C#] public virtual void GetObjectData(SerializationInfo info, StreamingContext
13 context);

14 [C++] public: virtual void GetObjectData(SerializationInfo* info,
15 StreamingContext context);

16 [VB] Overridable Public Sub GetObjectData(ByVal info As SerializationInfo,
17 ByVal context As StreamingContext)

18 [JScript] public function GetObjectData(info : SerializationInfo, context :
19 StreamingContext);

21 *Description*

22 Sets the **System.Runtime.Serialization.SerializationInfo** object with
23 information about the exception.

24 **System.TypeLoadException.GetObjectData(System.Runtime.Serializat**
25 **ion.SerializationInfo, System.Runtime.Serialization.StreamingContext)** sets a

System.Runtime.Serialization.SerializationInfo with all the exception object data targeted for serialization. During deserialization, the exception object is reconstituted from the **System.Runtime.Serialization.SerializationInfo** transmitted over the stream. The object that holds the serialized object data. The contextual information about the source or destination.

ToString

[C#] public override string ToString();

[C++] public: String* ToString();

[VB] Overrides Public Function ToString() As String

[JScript] public override function ToString() : String;

Description

Returns the fully qualified name of this exception and possibly the error message, the name of the inner exception, and the stack trace.

Return Value: The fully qualified class name, plus possibly the error message, the name of the inner exception, and the stack trace.

If there is no error message or if it is an empty string (""), then no error message is returned. The name of the inner exception and the stack trace are returned only if they are not **null**.

ExecutionEngineException class (System)

ToString

Description

The exception that is thrown when there is an internal error in the execution engine of the common language runtime. This class cannot be inherited.

System.ExecutionEngineException uses the HRESULT COR_E_EXECUTIONENGINE, which has the value 0x80131506.

ExecutionEngineException

Example Syntax:

ToString

[C#] public ExecutionEngineException();

[C++] public: ExecutionEngineException();

[VB] Public Sub New()

[JScript] public function ExecutionEngineException(); Initializes a new instance of the **System.ExecutionEngineException** class.

Description

Initializes a new instance of the **System.ExecutionEngineException** class with default properties.

The following table shows the initial property values for an instance of **System.ExecutionEngineException**.

ExecutionEngineException

Example Syntax:

ToString

[C#] public ExecutionEngineException(string message);

[C++] public: ExecutionEngineException(String* message);

1 [VB] Public Sub New(ByVal message As String)

2 [JScript] public function ExecutionEngineException(message : String);

3
4 *Description*

5 Initializes a new instance of the **System.ExecutionEngineException** class
6 with a specified error message.

7 The following table shows the initial property values for an instance of
8 **System.ExecutionEngineException** . The error message that explains the reason
9 for the exception.

10 ExecutionEngineException

11 *Example Syntax:*

12 ToString

13
14 [C#] public ExecutionEngineException(string message, Exception
15 innerException);

16 [C++] public: ExecutionEngineException(String* message, Exception*
17 innerException);

18 [VB] Public Sub New(ByVal message As String, ByVal innerException As
19 Exception)

20 [JScript] public function ExecutionEngineException(message : String,
21 innerException : Exception);

22
23 *Description*

1 Initializes a new instance of the **System.ExecutionEngineException** class
2 with a specified error message and a reference to the inner exception that is the
3 root cause of this exception.

4 When an **Exception** *X* is thrown as a direct result of a previous exception *Y*,
5 the **System.Exception.InnerException** property of *X* should contain a reference
6 to *Y*. The **InnerException** property returns the same value as was passed into the
7 constructor, or **null** if the inner exception value was not supplied to the
8 constructor. The error message that explains the reason for the exception. An
9 instance of **System.Exception** that is the cause of the current **Exception**. If
10 *innerException* is non-null, then the current **Exception** is raised in a catch block
11 handling *innerException*.

12 HelpLink

13 HResult

14 InnerException

15 Message

16 Source

17 StackTrace

18 TargetSite

19 FieldAccessException class (System)

20 ToString

23 *Description*

24 The exception that is thrown when there is an illegal attempt to access a
25 private or protected field inside a class.

System.FieldAccessException uses the HRESULT
COR_E_FIELDACCESS, which has the value 0x80131507.

FieldAccessException

Example Syntax:

ToString

[C#] public FieldAccessException();

[C++] public: FieldAccessException();

[VB] Public Sub New()

[JScript] public function FieldAccessException(); Initializes a new instance of the
System.FieldAccessException class.

Description

Initializes a new instance of the **System.FieldAccessException** class with
default properties.

The following table shows the initial property values for an instance of
System.FieldAccessException.

FieldAccessException

Example Syntax:

ToString

[C#] public FieldAccessException(string message);

[C++] public: FieldAccessException(String* message);

[VB] Public Sub New(ByVal message As String)

[JScript] public function FieldAccessException(message : String);

Description

Initializes a new instance of the **System.FieldAccessException** class with a specified error message.

The following table shows the initial property values for an instance of **System.FieldAccessException**. The error message that explains the reason for the exception.

FieldAccessException

Example Syntax:

ToString

[C#] protected FieldAccessException(SerializationInfo info, StreamingContext context);

[C++] protected: FieldAccessException(SerializationInfo* info, StreamingContext context);

[VB] Protected Sub New(ByVal info As SerializationInfo, ByVal context As StreamingContext)

[JScript] protected function FieldAccessException(info : SerializationInfo, context : StreamingContext);

Description

Initializes a new instance of the **System.FieldAccessException** class with serialized data.

This constructor is called during deserialization to reconstitute the exception object transmitted over a stream. For more information, see . The object

that holds the serialized object data. The contextual information about the source or destination.

FieldAccessException

Example Syntax:

ToString

[C#] public FieldAccessException(string message, Exception inner);

[C++] public: FieldAccessException(String* message, Exception* inner);

[VB] Public Sub New(ByVal message As String, ByVal inner As Exception)

[JScript] public function FieldAccessException(message : String, inner :
Exception);

Description

Initializes a new instance of the **System.FieldAccessException** class with a specified error message and a reference to the inner exception that is the root cause of this exception.

When an **Exception** *X* is thrown as a direct result of a previous exception *Y*, the **System.Exception.InnerException** property of *X* should contain a reference to *Y*. The **InnerException** property returns the same value as was passed into the constructor, or **null** if the inner exception value was not supplied to the constructor. The error message that explains the reason for the exception. An instance of **System.Exception** that is the cause of the current **Exception**. If *inner* is non-null, then the current **Exception** is raised in a catch block handling *inner*.

HelpLink

HResult

1 InnerException

2 Message

3 Source

4 StackTrace

5 TargetSite

6 FlagsAttribute class (System)

7 ToString

9
10 *Description*

11 Custom attribute indicating an enumeration should be treated as a bitfield;
12 that is, a set of flags.

13 Bitfields can be combined using a bitwise OR operation, whereas
14 enumerated constants cannot.

15 FlagsAttribute

16 *Example Syntax:*

17 ToString

18
19 [C#] public FlagsAttribute();

20 [C++] public: FlagsAttribute();

21 [VB] Public Sub New()

22 [JScript] public function FlagsAttribute();

23
24 *Description*

25 Initializes a new instance of the **FlagsAttribute** class.

1 TypeId

2 FormatException class (System)

3 ToString

4
5
6 *Description*

7 The exception that is thrown when the format of an argument does not meet
8 the parameter specifications of the invoked method.

9 **System.FormatException** is thrown when the format of an argument in a
10 method invocation does not match the format of the corresponding formal
11 parameter type. For example, if a method specifies a **System.String** parameter
12 consisting of two digits with an embedded period, passing a corresponding string
13 argument containing only two digits to that method would cause
14 **System.FormatException** to be thrown.

15 FormatException

16 *Example Syntax:*

17 ToString

18
19 [C#] public FormatException();

20 [C++] public: FormatException();

21 [VB] Public Sub New()

22 [JScript] public function FormatException(); Initializes a new instance of the

23 **System.FormatException** class.

24
25 *Description*

1 Initializes a new instance of the **System.FormatException** class with
2 default properties.

3 The following table shows the initial property values for an instance of
4 **System.FormatException** .

5 FormatException

6 *Example Syntax:*

7 ToString

9 [C#] public FormatException(string message);

10 [C++] public: FormatException(String* message);

11 [VB] Public Sub New(ByVal message As String)

12 [JScript] public function FormatException(message : String);

14 *Description*

15 Initializes a new instance of the **System.FormatException** class with a
16 specified error message.

17 The following table shows the initial property values for an instance of
18 **System.FormatException** . The error message that explains the reason for the
19 exception.

20 FormatException

21 *Example Syntax:*

22 ToString

24 [C#] protected FormatException(SerializationInfo info, StreamingContext
25 context);

1 [C++] protected: FormatException(SerializationInfo* info, StreamingContext
2 context);

3 [VB] Protected Sub New(ByVal info As SerializationInfo, ByVal context As
4 StreamingContext)

5 [JScript] protected function FormatException(info : SerializationInfo, context :
6 StreamingContext);

7 8 *Description*

9 Initializes a new instance of the **System.FormatException** class with
10 serialized data.

11 This constructor is called during deserialization to reconstitute the
12 exception object transmitted over a stream. For more information, see . The object
13 that holds the serialized object data. The contextual information about the source
14 or destination.

15 FormatException

16 *Example Syntax:*

17 ToString

18
19 [C#] public FormatException(string message, Exception innerException);

20 [C++] public: FormatException(String* message, Exception* innerException);

21 [VB] Public Sub New(ByVal message As String, ByVal innerException As
22 Exception)

23 [JScript] public function FormatException(message : String, innerException :
24 Exception);

Description

Initializes a new instance of the **System.FormatException** class with a specified error message and a reference to the inner exception that is the root cause of this exception.

When an **Exception** *X* is thrown as a direct result of a previous exception *Y*, the **System.Exception.InnerException** property of *X* should contain a reference to *Y*. The **InnerException** property returns the same value as was passed into the constructor, or **null** if the inner exception value was not supplied to the constructor. The error message that explains the reason for the exception. An instance of **System.Exception** that is the cause of the current **Exception**. If *innerException* is non-null, then the current **Exception** is raised in a catch block handling *innerException*.

HelpLink

HResult

InnerException

Message

Source

StackTrace

TargetSite

GC class (System)

ToString

Description

Controls the system garbage collector, a service that automatically reclaims unused memory.

Methods in this class influence when an object is garbage collected and when resources allocated by an object are released ("finalized"). Properties in this class provide information about the total amount of memory available in the system and the age category, or "generation", of memory allocated to an object.

MaxGeneration

ToString

[C#] public static int MaxGeneration {get;}

[C++] public: __property static int get_MaxGeneration();

[VB] Public Shared ReadOnly Property MaxGeneration As Integer

[JScript] public static function get MaxGeneration() : int;

Description

Gets the maximum number of generations the system currently supports.

A "generation" number indicates the relative age of each segment of allocated memory. The newest memory is in generation zero and the oldest memory is in generation **MaxGeneration**. The garbage collector service improves its performance by adjusting generation numbers each time it reclaims memory, then taking into consideration newer memory is more likely to be eligible for garbage collection than older memory.

Collect

[C#] public static void Collect();

1 [C++] public: static void Collect();

2 [VB] Public Shared Sub Collect()

3 [JScript] public static function Collect();

4
5 *Description*

6 Forces garbage collection of all generations.

7 Collect

8
9 [C#] public static void Collect(int generation);

10 [C++] public: static void Collect(int generation);

11 [VB] Public Shared Sub Collect(ByVal generation As Integer)

12 [JScript] public static function Collect(generation : int); Forces garbage collection.

13
14 *Description*

15 Forces garbage collection from generation zero through a specified
16 generation. The maximum generation to garbage collect.

17 GetGeneration

18
19 [C#] public static int GetGeneration(object obj);

20 [C++] public: static int GetGeneration(Object* obj);

21 [VB] Public Shared Function GetGeneration(ByVal obj As Object) As Integer

22 [JScript] public static function GetGeneration(obj : Object) : int; Returns the
23 current generation of an object.

24
25 *Description*

Returns the current generation of a specified object.

Return Value: The current generation of *obj* . The object for which generation information is retrieved.

GetGeneration

[C#] public static int GetGeneration(WeakReference wo);

[C++] public: static int GetGeneration(WeakReference* wo);

[VB] Public Shared Function GetGeneration(ByVal wo As WeakReference) As

Integer

[JScript] public static function GetGeneration(wo : WeakReference) : int;

Description

Returns the current generation of the target of a specified weak reference.

Return Value: The current generation of the target of *wo* . The weak reference of a target.

GetTotalMemory

[C#] public static long GetTotalMemory(bool forceFullCollection);

[C++] public: static __int64 GetTotalMemory(bool forceFullCollection);

[VB] Public Shared Function GetTotalMemory(ByVal forceFullCollection As Boolean) As Long

[JScript] public static function GetTotalMemory(forceFullCollection : Boolean) : long;

Description

Retrieves the number of bytes currently thought to be allocated. A parameter indicates whether this method should wait a short interval before returning while the system garbage collects and finalizes objects.

If *forceFullCollection* is **true**, this method waits a short interval before returning while the system garbage collects and finalizes objects. The duration of the interval is an internally specified limit determined by the number of garbage collection cycles completed and the change in the amount of memory recovered between cycles. A Boolean value, which if **true** indicates this method should wait before returning.

KeepAlive

[C#] public static void KeepAlive(object obj);

[C++] public: static void KeepAlive(Object* obj);

[VB] Public Shared Sub KeepAlive(ByVal obj As Object)

[JScript] public static function KeepAlive(obj : Object);

Description

References the specified object, making it ineligible for garbage collection from the start of the current routine to the point where this method is called.

When calling methods in unmanaged code (such as Win32 APIs, unmanaged DLLs, or methods using COM), it is sometimes necessary to indicate a particular object should not be garbage collected, even though there are no references to it from managed code or data. The object to reference.

ReRegisterForFinalize

```

1  [C#] public static void ReRegisterForFinalize(object obj);
2
3  [C++] public: static void ReRegisterForFinalize(Object* obj);
4
5  [VB] Public Shared Sub ReRegisterForFinalize(ByVal obj As Object)
6
7  [JScript] public static function ReRegisterForFinalize(obj : Object);
8

```

Description

Requests the system call the finalizer method for the specified object, for which **SuppressFinalize** has previously been called.

A finalizer can use this method to resurrect itself or an object it references. The object for which a finalizer should be called.

SuppressFinalize

```

14 [C#] public static void SuppressFinalize(object obj);
15
16 [C++] public: static void SuppressFinalize(Object* obj);
17
18 [VB] Public Shared Sub SuppressFinalize(ByVal obj As Object)
19
20 [JScript] public static function SuppressFinalize(obj : Object);
21

```

Description

Requests the system not call the finalizer method for the specified object. The object for which a finalizer should not be called.

WaitForPendingFinalizers

```

24 [C#] public static void WaitForPendingFinalizers();
25
26 [C++] public: static void WaitForPendingFinalizers();

```

1 [VB] Public Shared Sub WaitForPendingFinalizers()

2 [JScript] public static function WaitForPendingFinalizers();

3
4 *Description*

5 Suspends the current thread until the thread processing the queue of
6 finalizers has emptied that queue.

7 Finalizers are run on a separate thread of execution, so there is no guarantee
8 this method will terminate. However, this thread can be interrupted by another
9 thread while this method is in progress. This means you can start another thread
10 that waits for a period of time, then interrupts this thread if it is still suspended.

11 Guid structure (System)

12 WaitForPendingFinalizers

13
14
15 *Description*

16 Represents a globally unique identifier (GUID).

17 A GUID is a 128-bit integer (16 bytes) that can be used across all
18 computers and networks wherever a unique identifier is required. Such an
19 identifier has a very low probability of being duplicated.

20 WaitForPendingFinalizers

21
22 [C#] public static readonly Guid Empty;

23 [C++] public: static Guid Empty;

24 [VB] Public Shared ReadOnly Empty As Guid

25 [JScript] public static var Empty : Guid;

1
2 *Description*

3 Initializes a new instance of the **Guid** class.

4 Guid

5 *Example Syntax:*

6 WaitForPendingFinalizers

7
8 [C#] public Guid(byte[] b);

9 [C++] public: Guid(unsigned char b __gc[]);

10 [VB] Public Sub New(ByVal b() As Byte)

11 [JScript] public function Guid(b : Byte[]); Initializes a new instance of the **Guid**
12 class.

13
14 *Description*

15 Initializes a new instance of the **Guid** class using the specified array of
16 bytes. A 16 element byte array containing values with which to initialize the
17 GUID.

18 Guid

19 *Example Syntax:*

20 WaitForPendingFinalizers

21
22 [C#] public Guid(string g);

23 [C++] public: Guid(String* g);

24 [VB] Public Sub New(ByVal g As String)

25 [JScript] public function Guid(g : String);

Description

Initializes a new instance of the **Guid** class using the value represented by the specified string.

The string may begin and end with braces: '{', and '}'. A **String** that contains a GUID in the following format: hexadecimal digits are arranged in groups of 8, 4, 4, 4, and 12 digits with hyphens between the groups. The GUID can optionally be enclosed in matching braces. For example: dddddddd-dddd-dddd-dddd-dddddddddddd or {dddddddd-dddd-dddd-dddd-dddddddddddd}.

Guid

Example Syntax:

WaitForPendingFinalizers

```
[C#] public Guid(int a, short b, short c, byte[] d);
```

```
[C++] public: Guid(int a, short b, short c, unsigned char d __gc[]);
```

```
[VB] Public Sub New(ByVal a As Integer, ByVal b As Short, ByVal c As Short,
    ByVal d() As Byte)
```

```
[JScript] public function Guid(a : int, b : Int16, c : Int16, d : Byte[]);
```

Description

Initializes a new instance of the **Guid** class using the specified integers and byte array. The first 4 bytes of the GUID. The next 2 bytes of the GUID. The next 2 bytes of the GUID. The remaining 8 bytes of the GUID.

Guid

Example Syntax:

WaitForPendingFinalizers

```
[C#] public Guid(int a, short b, short c, byte d, byte e, byte f, byte g, byte h, byte i,  
byte j, byte k);
```

```
[C++] public: Guid(int a, short b, short c, unsigned char d, unsigned char e,  
unsigned char f, unsigned char g, unsigned char h, unsigned char i, unsigned char  
j, unsigned char k);
```

```
[VB] Public Sub New(ByVal a As Integer, ByVal b As Short, ByVal c As Short,  
ByVal d As Byte, ByVal e As Byte, ByVal f As Byte, ByVal g As Byte, ByVal h  
As Byte, ByVal i As Byte, ByVal j As Byte, ByVal k As Byte)
```

```
[JScript] public function Guid(a : int, b : Int16, c : Int16, d : Byte, e : Byte, f :  
Byte, g : Byte, h : Byte, i : Byte, j : Byte, k : Byte);
```

Description

Initializes a new instance of the **Guid** class using the specified integers and bytes.

Specifying individual bytes in this manner can be used to circumvent byte order restrictions ("big endian" or "little endian" byte order) on particular types of computers. The first 4 bytes of the GUID. The next 2 bytes of the GUID. The next 2 bytes of the GUID. The next byte of the GUID. The next byte of the GUID. The next byte of the GUID. The next byte of the GUID. The next byte of the GUID. The next byte of the GUID. The next byte of the GUID.

Guid

Example Syntax:

WaitForPendingFinalizers

```
[C#] public Guid(uint a, ushort b, ushort c, byte d, byte e, byte f, byte g, byte h,  
byte i, byte j, byte k);  
[C++] public: Guid(unsigned int a, unsigned short b, unsigned short c, unsigned  
char d, unsigned char e, unsigned char f, unsigned char g, unsigned char h,  
unsigned char i, unsigned char j, unsigned char k);  
[VB] Public Sub New(ByVal a As UInt32, ByVal b As UInt16, ByVal c As  
UInt16, ByVal d As Byte, ByVal e As Byte, ByVal f As Byte, ByVal g As Byte,  
ByVal h As Byte, ByVal i As Byte, ByVal j As Byte, ByVal k As Byte)  
[JScript] public function Guid(a : UInt32, b : UInt16, c : UInt16, d : Byte, e : Byte,  
f : Byte, g : Byte, h : Byte, i : Byte, j : Byte, k : Byte);
```

Description

Initializes a new instance of the **Guid** class using the specified unsigned integers and bytes.

Specifying the bytes in this manner avoids endianness issues. The first 4 bytes of the GUID. The next 2 bytes of the GUID. The next 2 bytes of the GUID. The next byte of the GUID. The next byte of the GUID. The next byte of the GUID. The next byte of the GUID. The next byte of the GUID. The next byte of the GUID. The next byte of the GUID.

CompareTo

```
[C#] public int CompareTo(object value);  
[C++] public: __sealed int CompareTo(Object* value);
```

[VB] NotOverridable Public Function CompareTo(ByVal value As Object) As Integer

[JScript] public function CompareTo(value : Object) : int;

Description

Compares this instance to a specified object and returns an indication of their relative values.

Return Value: A signed number indicating the relative values of this instance and *value*.

Any instance of **Guid**, regardless of its value, is considered greater than **null**. An object to compare, or **null**.

Equals

[C#] public override bool Equals(object o);

[C++] public: bool Equals(Object* o);

[VB] Overrides Public Function Equals(ByVal o As Object) As Boolean

[JScript] public override function Equals(o : Object) : Boolean;

Description

Returns a value indicating whether this instance is equal to a specified object.

Return Value: **true** if *o* is a **Guid** that has the same value as this instance; otherwise, **false**. The object to compare with this instance.

GetHashCode

1
2 [C#] public override int GetHashCode();

3 [C++] public: int GetHashCode();

4 [VB] Overrides Public Function GetHashCode() As Integer

5 [JScript] public override function GetHashCode() : int;

6
7 *Description*

8 Returns the hash code for this instance.

9 *Return Value:* The hash code for this instance.

10 **NewGuid**

11
12 [C#] public static Guid NewGuid();

13 [C++] public: static Guid NewGuid();

14 [VB] Public Shared Function NewGuid() As Guid

15 [JScript] public static function NewGuid() : Guid;

16
17 *Description*

18 Initializes a new instance of the **Guid** class.

19 *Return Value:* A new **Guid** object.

20 This is a convenient **static** method that you can call to get a new **Guid** .

21 **op_Equality**

22
23 [C#] public static bool operator ==(Guid a, Guid b);

24 [C++] public: static bool op_Equality(Guid a, Guid b);

25 [VB] returnValue = Guid.op_Equality(a, b)

1 [JScript] returnValue = a == b;

3 *Description*

4 Returns an indication whether the values of two specified **Guid** objects are
5 equal.

6 *Return Value:* **true** if *a* and *b* are equal; otherwise, **false** . A **Guid** object. A **Guid**
7 object.

8 op_Inequality

10 [C#] public static bool operator !=(Guid a, Guid b);

11 [C++] public: static bool op_Inequality(Guid a, Guid b);

12 [VB] returnValue = Guid.op_Inequality(a, b)

13 [JScript] returnValue = a != b;

15 *Description*

16 Returns an indication whether the values of two specified **Guid** objects are
17 not equal.

18 *Return Value:* **true** if *a* and *b* are not equal; otherwise, **false** . A **Guid** object. A
19 **Guid** object.

20 ToByteArray

22 [C#] public byte[] ToByteArray();

23 [C++] public: unsigned char ToByteArray() __gc[];

24 [VB] Public Function ToByteArray() As Byte()

25 [JScript] public function ToByteArray() : Byte[];

Description

Returns a 16-element byte array that contains the value of the GUID.

Return Value: A 16-element byte array.

ToString

[C#] public override string ToString();

[C++] public: String* ToString();

[VB] Overrides Public Function ToString() As String

[JScript] public override function ToString() : String; Returns a **String** representation of the value of this instance of the **Guid** class.

Description

Returns a **String** representation of the value of this instance in Registry format.

Return Value: A **String** formatted in this pattern: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx where the value of the GUID is represented as a series of lower-case hexadecimal digits in groups of 8, 4, 4, 4, and 12 digits and separated by hyphens. An example of a return value is "382c74c3-721d-4f34-80e5-57657b6cbc27".

This method provides a default GUID format that is sufficient for typical use; however, other versions of this method that take a format parameter provide a few common format variations.

ToString

```

1
2 [C#] public string ToString(string format);
3 [C++] public: String* ToString(String* format);
4 [VB] Public Function ToString(ByVal format As String) As String
5 [JScript] public function ToString(format : String) : String;
6

```

7 *Description*

8 Returns a **String** representation of the value of this **Guid** instance,
9 according to the provided format specifier.

10 *Return Value:* A **System.String** representation of the value of this **Guid** instance.

11 *format* can contain the following format specifiers. In the table that follows,
12 all digits in the return value are hexadecimal. Each character 'x' represents a
13 hexadecimal digit; each hyphen ('-'), bracket ('{', '}'), and parenthesis ('(', ')')
14 appears as shown. A **String** containing a single format specifier character
15 indicating how the GUID value should be formatted.

16 *ToString*

```

17
18 [C#] public string ToString(string format, IFormatProvider provider);
19 [C++] public: __sealed String* ToString(String* format, IFormatProvider*
20 provider);
21 [VB] NotOverridable Public Function ToString(ByVal format As String, ByVal
22 provider As IFormatProvider) As String
23 [JScript] public function ToString(format : String, provider : IFormatProvider) :
24 String;
25

```


Description

Returns a **String** representation of the value of this instance of the **Guid** class, according to the provided format specifier and culture-specific format information.

Return Value: A **System.String** representation of the value of this **Guid** instance.

format can contain the following format specifiers. In the table that follows, all digits in the return value are hexadecimal. Each character 'x' represents a hexadecimal digit; each hyphen ('-'), bracket ('{', '}'), and parenthesis ('(', ')') appears as shown. A **String** containing a single format specifier character indicating how the GUID value should be formatted. (Reserved) An **IFormatProvider** reference that supplies culture-specific formatting services.

IAppDomainSetup interface (System)

ToString

Description

ApplicationBase

ToString

[C#] string ApplicationBase {get; set;}

[C++] String* get_ApplicationBase();void set_ApplicationBase(String*);

[VB] Property ApplicationBase As String

[JScript] abstract function get ApplicationBase() : String;public abstract function

1 set ApplicationBase(String);

2
3 *Description*

4
5 ApplicationName

6 ToString

7
8 [C#] string ApplicationName {get; set;}

9 [C++] String* get_ApplicationName();void set_ApplicationName(String*);

10 [VB] Property ApplicationName As String

11 [JScript] abstract function get ApplicationName() : String;public abstract function

12 set ApplicationName(String);

13
14 *Description*

15
16 CachePath

17 ToString

18
19 [C#] string CachePath {get; set;}

20 [C++] String* get_CachePath();void set_CachePath(String*);

21 [VB] Property CachePath As String

22 [JScript] abstract function get CachePath() : String;public abstract function set

23 CachePath(String);

24
25 *Description*

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

ConfigurationFile

ToString

```
[C#] string ConfigurationFile {get; set;}
[C++] String* get_ConfigurationFile();void set_ConfigurationFile(String*);
[VB] Property ConfigurationFile As String
[JScript] abstract function get ConfigurationFile() : String;public abstract function
set ConfigurationFile(String);
```

Description

DynamicBase

ToString

```
[C#] string DynamicBase {get; set;}
[C++] String* get_DynamicBase();void set_DynamicBase(String*);
[VB] Property DynamicBase As String
[JScript] abstract function get DynamicBase() : String;public abstract function set
DynamicBase(String);
```

Description

LicenseFile

ToString

1
2 [C#] string LicenseFile {get; set;}

3 [C++] String* get_LicenseFile();void set_LicenseFile(String*);

4 [VB] Property LicenseFile As String

5 [JScript] abstract function get LicenseFile() : String;public abstract function set
6 LicenseFile(String);

7
8 *Description*

9
10 PrivateBinPath

11 ToString

12
13 [C#] string PrivateBinPath {get; set;}

14 [C++] String* get_PrivateBinPath();void set_PrivateBinPath(String*);

15 [VB] Property PrivateBinPath As String

16 [JScript] abstract function get PrivateBinPath() : String;public abstract function set
17 PrivateBinPath(String);

18
19 *Description*

20
21 PrivateBinPathProbe

22 ToString

23
24 [C#] string PrivateBinPathProbe {get; set;}

25 [C++] String* get_PrivateBinPathProbe();void set_PrivateBinPathProbe(String*);

1 [VB] Property PrivateBinPathProbe As String

2 [JScript] abstract function get PrivateBinPathProbe() : String;public abstract
3 function set PrivateBinPathProbe(String);

4
5 *Description*

6
7 ShadowCopyDirectories

8 ToString

9
10 [C#] string ShadowCopyDirectories {get; set;}

11 [C++] String* get_ShadowCopyDirectories();void
12 set_ShadowCopyDirectories(String*);

13 [VB] Property ShadowCopyDirectories As String

14 [JScript] abstract function get ShadowCopyDirectories() : String;public abstract
15 function set ShadowCopyDirectories(String);

16
17 *Description*

18
19 ShadowCopyFiles

20 ToString

21
22 [C#] string ShadowCopyFiles {get; set;}

23 [C++] String* get_ShadowCopyFiles();void set_ShadowCopyFiles(String*);

24 [VB] Property ShadowCopyFiles As String

25 [JScript] abstract function get ShadowCopyFiles() : String;public abstract function

1 set ShadowCopyFiles(String);

2
3 *Description*

4
5 IAsyncResult interface (System)

6 ToString

7
8
9 *Description*

10 Represents the the status on an asynchronous operation.

11 The **System.IAsyncResult** interface is implemented by classes containing
12 methods that can operate asynchronously. It is the return type of the "BeginXXX"
13 method that initiates an asynchronous operation, and is the type of the third
14 parameter of the "EndXXX" method that concludes an asynchronous operation.

15 AsyncState

16 ToString

17
18 [C#] object AsyncState {get;}

19 [C++] Object* get_AsyncState();

20 [VB] ReadOnly Property AsyncState As Object

21 [JScript] abstract function get AsyncState() : Object;

22
23 *Description*

24 Gets a user-defined object that qualifies or contains information about an
25 asynchronous operation.

This property returns the object that is the last parameter of the "BeginXXX" method that initiates an asynchronous operation.

AsyncWaitHandle

ToString

[C#] WaitHandle AsyncWaitHandle {get;}

[C++] WaitHandle* get_AsyncWaitHandle();

[VB] ReadOnly Property AsyncWaitHandle As WaitHandle

[JScript] abstract function get AsyncWaitHandle() : WaitHandle;

Description

Gets a **System.Threading.WaitHandle** instance that is used to wait for an asynchronous operation to complete.

The return value enables the client to wait for an asynchronous operation to complete instead of polling **System.IAsyncResult.IsCompleted** until the operation concludes. The return value can be used to perform a **System.Threading.WaitHandle.WaitOne(System.Int32,System.Boolean)** , **System.Threading.WaitHandle.WaitAny(System.Threading.WaitHandle[],System.Int32,System.Boolean)** , or **System.Threading.WaitHandle.WaitAll(System.Threading.WaitHandle[],System.Int32,System.Boolean)** operation.

CompletedSynchronously

ToString

[C#] bool CompletedSynchronously {get;}

1 [C++] bool get_CompletedSynchronously();

2 [VB] ReadOnly Property CompletedSynchronously As Boolean

3 [JScript] abstract function get CompletedSynchronously() : Boolean;

4
5 *Description*

6 Gets an indication whether the "BeginXXX" call completed synchronously.

7 If the synchronous completion of the call is detected in the

8 **System.AsyncCallback** delegate, it is probable that the thread that called

9 "BeginXXX" is the current thread. Most implementers of the

10 **System.IAsyncResult** interface will not use this ability and will return **false** .

11 IsCompleted

12 ToString

13
14 [C#] bool IsCompleted {get;}

15 [C++] bool get_IsCompleted();

16 [VB] ReadOnly Property IsCompleted As Boolean

17 [JScript] abstract function get IsCompleted() : Boolean;

18
19 *Description*

20 Gets an indication whether the asynchronous operation is finished.

21 Implementers will typically return the value of a private field or internal

22 test as the value of this property.

23 ICloneable interface (System)

24 ToString

1
2
3 *Description*

4 Supports cloning, which creates a new instance of a class with the same
5 value as an existing instance.

6 The **System.ICloneable** interface contains one member,
7 **System.ICloneable.Clone** , which is intended to support cloning beyond that
8 supplied by **System.Object.MemberwiseClone** .

9 Clone

10
11 [C#] object Clone();

12 [C++] Object* Clone();

13 [VB] Function Clone() As Object

14 [JScript] function Clone() : Object;

15
16 *Description*

17 Creates a new object that is a copy of the current instance.

18 *Return Value:* A new object that is a copy of this instance.

19 **System.ICloneable.Clone** can be implemented either as a deep copy or a
20 shallow copy. In a deep copy, all objects are duplicated; whereas, in a shallow
21 copy, only the top-level objects are duplicated and the lower levels contain
22 references.

23 IComparable interface (System)

24 Clone

Description

Defines a generalized comparison method, which a value type or class implements to create a type-specific comparison method.

This interface is implemented by types whose values can be ordered; for example, the numeric and string classes.

CompareTo

[C#] int CompareTo(object obj);

[C++] int CompareTo(Object* obj);

[VB] Function CompareTo(ByVal obj As Object) As Integer

[JScript] function CompareTo(obj : Object) : int;

Description

Compares the current instance with another object of the same type.

Return Value: A 32-bit signed integer that indicates the relative order of the comparands. The return value has these meanings: Value Meaning Less than zero

This instance is less than *obj*.

This method is only a definition and must be implemented by a specific class or value type to have effect. The meaning of the comparisons, "less than," "equal to," and "greater than," depends on the particular implementation. An object to compare with this instance.

IConvertible interface (System)

CompareTo

1
2
3 *Description*

4 Defines methods that convert the value of the implementing reference or
5 value type to a common language runtime type that has an equivalent value.

6 This interface provides methods to convert the value of an instance of an
7 implementing type to a common language runtime type that has an equivalent
8 value. The common language runtime types are **System.Boolean** , **System.SByte** ,
9 **System.Byte** , **System.Int16** , **System.UInt16** , **System.Int32** , **System.UInt32** ,
10 **System.Int64** , **System.UInt64** , **System.Single** , **System.Double** ,
11 **System.Decimal** , **System.DateTime** , **System.Char** , and **System.String** .

12 **GetTypeCode**

13
14 [C#] **TypeCode** GetTypeCode();
15 [C++] **TypeCode** GetTypeCode();
16 [VB] **Function** GetTypeCode() **As** **TypeCode**
17 [JScript] **function** GetTypeCode() : **TypeCode**;

18
19 *Description*

20 Returns the **System.TypeCode** for this instance.

21 *Return Value:* The enumerated constant that is the **System.TypeCode** of the class
22 or value type that implements this interface.

23 **ToBoolean**

24
25 [C#] **bool** ToBoolean(IFormatProvider provider);

1 [C++] bool ToBoolean(IFormatProvider* provider);

2 [VB] Function ToBoolean(ByVal provider As IFormatProvider) As Boolean

3 [JScript] function ToBoolean(provider : IFormatProvider) : Boolean;

4
5 *Description*

6 Converts the value of this instance to an equivalent Boolean value using the
7 specified culture-specific formatting information.

8 *Return Value:* A Boolean value equivalent to the value of this instance. An
9 **System.IFormatProvider** interface implementation that supplies culture-specific
10 formatting information.

11 *ToByte*

12
13 [C#] byte ToByte(IFormatProvider provider);

14 [C++] unsigned char ToByte(IFormatProvider* provider);

15 [VB] Function ToByte(ByVal provider As IFormatProvider) As Byte

16 [JScript] function ToByte(provider : IFormatProvider) : Byte;

17
18 *Description*

19 Converts the value of this instance to an equivalent 8-bit unsigned integer
20 using the specified culture-specific formatting information.

21 *Return Value:* An 8-bit unsigned integer equivalent to the value of this instance.

22 An **System.IFormatProvider** interface implementation that supplies culture-
23 specific formatting information.

24 *ToChar*

[C#] char ToChar(IFormatProvider provider);
[C++] __wchar_t ToChar(IFormatProvider* provider);
[VB] Function ToChar(ByVal provider As IFormatProvider) As Char
[JScript] function ToChar(provider : IFormatProvider) : Char;

Description

Converts the value of this instance to an equivalent Unicode character using the specified culture-specific formatting information.

Return Value: A Unicode character equivalent to the value of this instance. An **System.IFormatProvider** interface implementation that supplies culture-specific formatting information.

ToDateTime

[C#] DateTime ToDateTime(IFormatProvider provider);
[C++] DateTime ToDateTime(IFormatProvider* provider);
[VB] Function ToDateTime(ByVal provider As IFormatProvider) As DateTime
[JScript] function ToDateTime(provider : IFormatProvider) : DateTime;

Description

Converts the value of this instance to an equivalent **System.DateTime** using the specified culture-specific formatting information.

Return Value: A **System.DateTime** instance equivalent to the value of this instance. An **System.IFormatProvider** interface implementation that supplies culture-specific formatting information.

ToDecimal

[C#] decimal ToDecimal(IFormatProvider provider);
[C++] Decimal ToDecimal(IFormatProvider* provider);
[VB] Function ToDecimal(ByVal provider As IFormatProvider) As Decimal
[JScript] function ToDecimal(provider : IFormatProvider) : Decimal;

Description

Converts the value of this instance to an equivalent **System.Decimal** number using the specified culture-specific formatting information.

Return Value: A **System.Decimal** number equivalent to the value of this instance.

An **System.IFormatProvider** interface implementation that supplies culture-specific formatting information.

ToDouble

[C#] double ToDouble(IFormatProvider provider);
[C++] double ToDouble(IFormatProvider* provider);
[VB] Function ToDouble(ByVal provider As IFormatProvider) As Double
[JScript] function ToDouble(provider : IFormatProvider) : double;

Description

Converts the value of this instance to an equivalent double-precision floating-point number using the specified culture-specific formatting information.

Return Value: A double-precision floating-point number equivalent to the value of

1 this instance. An **System.IFormatProvider** interface implementation that supplies
2 culture-specific formatting information.

3 ToInt16

4
5 [C#] short ToInt16(IFormatProvider provider);

6 [C++] short ToInt16(IFormatProvider* provider);

7 [VB] Function ToInt16(ByVal provider As IFormatProvider) As Short

8 [JScript] function ToInt16(provider : IFormatProvider) : Int16;

9 10 *Description*

11 Converts the value of this instance to an equivalent 16-bit signed integer
12 using the specified culture-specific formatting information.

13 *Return Value:* An 16-bit signed integer equivalent to the value of this instance. An
14 **System.IFormatProvider** interface implementation that supplies culture-specific
15 formatting information.

16 ToInt32

17
18 [C#] int ToInt32(IFormatProvider provider);

19 [C++] int ToInt32(IFormatProvider* provider);

20 [VB] Function ToInt32(ByVal provider As IFormatProvider) As Integer

21 [JScript] function ToInt32(provider : IFormatProvider) : int;

22 23 *Description*

24 Converts the value of this instance to an equivalent 32-bit signed integer
25 using the specified culture-specific formatting information.

Return Value: An 32-bit signed integer equivalent to the value of this instance. An **System.IFormatProvider** interface implementation that supplies culture-specific formatting information.

ToInt64

[C#] long ToInt64(IFormatProvider provider);

[C++] __int64 ToInt64(IFormatProvider* provider);

[VB] Function ToInt64(ByVal provider As IFormatProvider) As Long

[JScript] function ToInt64(provider : IFormatProvider) : long;

Description

Converts the value of this instance to an equivalent 64-bit signed integer using the specified culture-specific formatting information.

Return Value: An 64-bit signed integer equivalent to the value of this instance. An **System.IFormatProvider** interface implementation that supplies culture-specific formatting information.

ToSByte

[C#] sbyte ToSByte(IFormatProvider provider);

[C++] char ToSByte(IFormatProvider* provider);

[VB] Function ToSByte(ByVal provider As IFormatProvider) As SByte

[JScript] function ToSByte(provider : IFormatProvider) : SByte;

Description

Converts the value of this instance to an equivalent 8-bit signed integer using the specified culture-specific formatting information.

Return Value: An 8-bit signed integer equivalent to the value of this instance. An **System.IFormatProvider** interface implementation that supplies culture-specific formatting information.

ToSingle

[C#] float ToSingle(IFormatProvider provider);

[C++] float ToSingle(IFormatProvider* provider);

[VB] Function ToSingle(ByVal provider As IFormatProvider) As Single

[JScript] function ToSingle(provider : IFormatProvider) : float;

Description

Converts the value of this instance to an equivalent single-precision floating-point number using the specified culture-specific formatting information.

Return Value: A single-precision floating-point number equivalent to the value of this instance. An **System.IFormatProvider** interface implementation that supplies culture-specific formatting information.

ToString

[C#] string ToString(IFormatProvider provider);

[C++] String* ToString(IFormatProvider* provider);

[VB] Function ToString(ByVal provider As IFormatProvider) As String

[JScript] function ToString(provider : IFormatProvider) : String;

Description

Converts the value of this instance to an equivalent **System.String** using the specified culture-specific formatting information.

Return Value: A **System.String** instance equivalent to the value of this instance.

An **System.IFormatProvider** interface implementation that supplies culture-specific formatting information.

ToType

[C#] object ToType(Type conversionType, IFormatProvider provider);

[C++] Object* ToType(Type* conversionType, IFormatProvider* provider);

[VB] Function ToType(ByVal conversionType As Type, ByVal provider As IFormatProvider) As Object

[JScript] function ToType(conversionType : Type, provider : IFormatProvider) : Object;

Description

Converts the value of this instance to an **System.Object** of the specified **System.Type** that has an equivalent value, using the specified culture-specific formatting information.

Return Value: An **System.Object** instance of type *conversionType* whose value is equivalent to the value of this instance. The **System.Type** to which the value of this instance is converted. An **System.IFormatProvider** interface implementation that supplies culture-specific formatting information.

ToUInt16

1
2 [C#] ushort ToUInt16(IFormatProvider provider);

3 [C++] unsigned short ToUInt16(IFormatProvider* provider);

4 [VB] Function ToUInt16(ByVal provider As IFormatProvider) As UInt16

5 [JScript] function ToUInt16(provider : IFormatProvider) : UInt16;

6
7 *Description*

8 Converts the value of this instance to an equivalent 16-bit unsigned integer
9 using the specified culture-specific formatting information.

10 *Return Value:* An 16-bit unsigned integer equivalent to the value of this instance.

11 An **System.IFormatProvider** interface implementation that supplies culture-
12 specific formatting information.

13 ToUInt32

14
15 [C#] uint ToUInt32(IFormatProvider provider);

16 [C++] unsigned int ToUInt32(IFormatProvider* provider);

17 [VB] Function ToUInt32(ByVal provider As IFormatProvider) As UInt32

18 [JScript] function ToUInt32(provider : IFormatProvider) : UInt32;

19
20 *Description*

21 Converts the value of this instance to an equivalent 32-bit unsigned integer
22 using the specified culture-specific formatting information.

23 *Return Value:* An 32-bit unsigned integer equivalent to the value of this instance.

24 An **System.IFormatProvider** interface implementation that supplies culture-
25 specific formatting information.

ToUInt64

[C#] ulong ToUInt64(IFormatProvider provider);

[C++] unsigned __int64 ToUInt64(IFormatProvider* provider);

[VB] Function ToUInt64(ByVal provider As IFormatProvider) As UInt64

[JScript] function ToUInt64(provider : IFormatProvider) : UInt64;

Description

Converts the value of this instance to an equivalent 64-bit unsigned integer using the specified culture-specific formatting information.

Return Value: An 64-bit unsigned integer equivalent to the value of this instance.

An **System.IFormatProvider** interface implementation that supplies culture-specific formatting information.

ICustomFormatter interface (System)

ToUInt64

Description

Defines a method that supports custom, user-defined formatting of the value of an object.

When this interface is implemented by a reference or value type, the **System.ICustomFormatter.Format(System.String,System.Object,System.IFormatProvider)** method returns a custom-formatted string representation of an object's value.

Format

[C#] string Format(string format, object arg, IFormatProvider formatProvider);

[C++] String* Format(String* format, Object* arg, IFormatProvider*
formatProvider);

[VB] Function Format(ByVal format As String, ByVal arg As Object, ByVal
formatProvider As IFormatProvider) As String

[JScript] function Format(format : String, arg : Object, formatProvider :
IFormatProvider) : String;

Description

Converts the value of a specified object to an equivalent string
representation using specified format and culture-specific formatting information.

Return Value: The string representation of the value of *arg* , formatted as specified
by *format* and *formatProvider* .

The *format* parameter contains a user-defined formatting specification. For
more information about standard .NET Framework formatting specifications, see .

A format string containing formatting specifications. An object to format. An
System.IFormatProvider object that supplies format information about the
current instance.

IDisposable interface (System)

Format

Description

Defines a method to release allocated unmanaged resources.

The common language runtime garbage collector automatically releases memory allocated to a managed object when that object is no longer used. Furthermore, it is unpredictable when garbage collection will occur. However, the garbage collector has no knowledge of unmanaged resources, such as window handles and open files and streams.

Dispose

[C#] void Dispose();

[C++] void Dispose();

[VB] Sub Dispose()

[JScript] function Dispose();

Description

Releases unmanaged resources.

Use this method to close or release unmanaged resources such as files, streams, and handles, held by an instance of the class that implements this interface.

IFormatProvider interface (System)

Dispose

Description

Provides a mechanism for retrieving an object to control formatting.

A class or value type implements the

System.IFormatProvider.GetFormat(System.Type) method of this interface to

1 obtain an object that provides format information or processing for the
2 implementing type.

3 GetFormat

4
5 [C#] object GetFormat(Type formatType);

6 [C++] Object* GetFormat(Type* formatType);

7 [VB] Function GetFormat(ByVal formatType As Type) As Object

8 [JScript] function GetFormat(formatType : Type) : Object;

9
10 *Description*

11 Gets the format object of the specified type.

12 *Return Value:* A format object of type *formatType* -or- **null** if there is no format
13 object of type *formatType* . An object that specifies the type of format object to
14 get.

15 IFormattable interface (System)

16 GetFormat

17
18
19 *Description*

20 Provides functionality to format the value of an object into a string
21 representation.

22 **System.IFormattable** is implemented by the base data types.

23 ToString

24
25 [C#] string ToString(string format, IFormatProvider formatProvider);

1 [C++] String* ToString(String* format, IFormatProvider* formatProvider);

2 [VB] Function ToString(ByVal format As String, ByVal formatProvider As

3 IFormatProvider) As String

4 [JScript] function ToString(format : String, formatProvider : IFormatProvider) :

5 String;

6
7 *Description*

8 Formats the value of the current instance using the specified format.

9 *Return Value:* A **System.String** containing the value of the current instance in the
10 specified format.

11 **System.Globalization.NumberFormatInfo** ,

12 **System.Globalization.DateTimeFormatInfo** and

13 **System.Globalization.CultureInfo** implement the **System.IFormatProvider**

14 interface. The **System.String** specifying the format to use. The

15 **System.IFormatProvider** to use to format the value.

16 IndexOutOfRangeException class (System)

17 ToString

18
19
20 *Description*

21 The exception that is thrown when an attempt is made to access an element
22 of an array with an index that is outside the bounds of the array. This class cannot
23 be inherited.

24 **System.IndexOutOfRangeException** uses the HRESULT

25 COR_E_INDEXOUTOFRANGE, which has the value 0x80131508.

IndexOutOfRangeException

Example Syntax:

ToString

```

[C#] public IndexOutOfRangeException();
[C++] public: IndexOutOfRangeException();
[VB] Public Sub New()
[JScript] public function IndexOutOfRangeException();
Initializes a new instance
of the System.IndexOutOfRangeException class.

```

Description

Initializes a new instance of the **System.IndexOutOfRangeException** class with default properties.

The following table shows the initial property values for an instance of **System.IndexOutOfRangeException** .

IndexOutOfRangeException

Example Syntax:

ToString

```

[C#] public IndexOutOfRangeException(string message);
[C++] public: IndexOutOfRangeException(String* message);
[VB] Public Sub New(ByVal message As String)
[JScript] public function IndexOutOfRangeException(message : String);

```

Description

1 Initializes a new instance of the **System.IndexOutOfRangeException**
2 class with a specified error message.

3 The following table shows the initial property values for an instance of
4 **System.IndexOutOfRangeException** . The error message that explains the
5 reason for the exception.

6 IndexOutOfRangeException

7 *Example Syntax:*

8 ToString

9
10 [C#] public IndexOutOfRangeException(string message, Exception
11 innerException);

12 [C++] public: IndexOutOfRangeException(String* message, Exception*
13 innerException);

14 [VB] Public Sub New(ByVal message As String, ByVal innerException As
15 Exception)

16 [JScript] public function IndexOutOfRangeException(message : String,
17 innerException : Exception);

18
19 *Description*

20 Initializes a new instance of the **System.IndexOutOfRangeException**
21 class with a specified error message and a reference to the inner exception that is
22 the root cause of this exception.

23 When an **Exception** *X* is thrown as a direct result of a previous exception *Y* ,
24 the **System.Exception.InnerException** property of *X* should contain a reference
25 to *Y* . The **InnerException** property returns the same value as was passed into the

1 constructor, or **null** if the inner exception value was not supplied to the
2 constructor. The error message that explains the reason for the exception. An
3 instance of **System.Exception** that is the cause of the current **Exception**. If
4 *innerException* is non-null, then the current **Exception** is raised in a catch block
5 handling *innerException* .

6 HelpLink

7 HRESULT

8 InnerException

9 Message

10 Source

11 StackTrace

12 TargetSite

13 Int16 structure (System)

14 ToString

17 *Description*

18 Represents a 16-bit signed integer.

19 The **Int16** value type represents signed integers with values ranging from
20 negative 32768 through positive 32767.

21 ToString

22
23 [C#] public const short MaxValue;

24 [C++] public: const short MaxValue;

25 [VB] Public Const MaxValue As Short

1 [JScript] public var MaxValue : Int16;

3 *Description*

4 A constant representing the largest possible value of **Int16** .

5 The value of this constant is 32767; that is, hexadecimal 0x7FFF.

6 ToString

8 [C#] public const short MinValue;

9 [C++] public: const short MinValue;

10 [VB] Public Const MinValue As Short

11 [JScript] public var MinValue : Int16;

13 *Description*

14 A constant representing the smallest possible value of **Int16** .

15 The value of this constant is -32768; that is, hexadecimal 0x8000.

16 CompareTo

18 [C#] public int CompareTo(object value);

19 [C++] public: __sealed int CompareTo(Object* value);

20 [VB] NotOverridable Public Function CompareTo(ByVal value As Object) As

21 Integer

22 [JScript] public function CompareTo(value : Object) : int;

24 *Description*

1 Compares this instance to a specified object and returns an indication of
2 their relative values.

3 *Return Value:* A signed number indicating the relative values of this instance and
4 *value* .

5 An **Int16** , regardless of its value, is considered greater than a null
6 reference. An object to compare, or **null**.

7 Equals

8
9 [C#] public override bool Equals(object obj);

10 [C++] public: bool Equals(Object* obj);

11 [VB] Overrides Public Function Equals(ByVal obj As Object) As Boolean

12 [JScript] public override function Equals(obj : Object) : Boolean;

13 14 Description

15 Returns a value indicating whether this instance is equal to a specified
16 object.

17 *Return Value:* **true** if *obj* is an instance of **Int16** and equals the value of this
18 instance; otherwise, **false** . An object to compare with this instance.

19 GetHashCode

20
21 [C#] public override int GetHashCode();

22 [C++] public: int GetHashCode();

23 [VB] Overrides Public Function GetHashCode() As Integer

24 [JScript] public override function GetHashCode() : int;

1
2 *Description*

3 Returns the hash code for this instance.

4 *Return Value:* A 32-bit signed integer hash code.

5 **GetTypeCode**

6
7 [C#] public TypeCode GetTypeCode();

8 [C++] public: __sealed TypeCode GetTypeCode();

9 [VB] NotOverridable Public Function GetTypeCode() As TypeCode

10 [JScript] public function GetTypeCode() : TypeCode;

11
12 *Description*

13 Returns the **TypeCode** for value type **Int16** .

14 *Return Value:* The enumerated constant, **System.TypeCode.Int16** .

15 **Parse**

16
17 [C#] public static short Parse(string s);

18 [C++] public: static short Parse(String* s);

19 [VB] Public Shared Function Parse(ByVal s As String) As Short

20 [JScript] public static function Parse(s : String) : Int16; Converts the **String**
21 representation of a number to its 16-bit signed integer equivalent.

22
23 *Description*

Converts the **String** representation of a number to its 16-bit signed integer equivalent.

Return Value: A 16-bit signed integer equivalent to the number contained in *s* .

s contains a number of the form: [ws][sign]digits[ws] Items in square brackets ('[' and ']') are optional, and other items are as follows. A **System.String** containing a number to convert.

Parse

[C#] public static short Parse(string s, IFormatProvider provider);

[C++] public: static short Parse(String* s, IFormatProvider* provider);

[VB] Public Shared Function Parse(ByVal s As String, ByVal provider As IFormatProvider) As Short

[JScript] public static function Parse(s : String, provider : IFormatProvider) :

Int16;

Description

Converts the **String** representation of a number in a specified culture-specific format to its 16-bit signed integer equivalent.

Return Value: A 16-bit signed integer equivalent to the number specified in *s* .

s contains a number of the form: [ws][sign]digits[ws] Items in square brackets ('[' and ']') are optional, and other items are as follows. A **System.String** containing a number to convert. An **System.IFormatProvider** interface implementation which supplies culture-specific formatting information about *s*.

Parse

[C#] public static short Parse(string s, NumberStyles style);

[C++] public: static short Parse(String* s, NumberStyles style);

[VB] Public Shared Function Parse(ByVal s As String, ByVal style As NumberStyles) As Short

[JScript] public static function Parse(s : String, style : NumberStyles) : Int16;

Description

Converts the **String** representation of a number in a specified style to its 16-bit signed integer equivalent.

Return Value: A 16-bit signed integer equivalent to the number specified in *s*.

s contains a number of the form: [ws][sign]digits[ws] Items in square brackets ('[' and ']') are optional, and other items are as follows. A **System.String** containing a number to convert. The combination of one or more **System.Globalization.NumberStyles** constants that indicate the permitted format of *s*.

Parse

[C#] public static short Parse(string s, NumberStyles style, IFormatProvider provider);

[C++] public: static short Parse(String* s, NumberStyles style, IFormatProvider* provider);

[VB] Public Shared Function Parse(ByVal s As String, ByVal style As NumberStyles, ByVal provider As IFormatProvider) As Short

[JScript] public static function Parse(s : String, style : NumberStyles, provider :

1 IFormatProvider) : Int16;

3 *Description*

4 Converts the **String** representation of a number in a specified style and
5 culture-specific format to its 16-bit signed integer equivalent.

6 *Return Value:* A 16-bit signed integer equivalent to the number specified in *s* .

7 *s* contains a number of the form: [ws][sign]digits[ws] Items in square
8 brackets ('[' and ']') are optional, and other items are as follows. A **System.String**
9 containing a number to convert. The combination of one or more
10 **System.Globalization.NumberStyles** constants that indicate the permitted format
11 of *s*. An **System.IFormatProvider** interface implementation which supplies
12 culture-specific formatting information about *s*.

13 IConvertible.ToBoolean

15 [C#] bool IConvertible.ToBoolean(IFormatProvider provider);

16 [C++] bool IConvertible::ToBoolean(IFormatProvider* provider);

17 [VB] Function ToBoolean(ByVal provider As IFormatProvider) As Boolean

18 Implements IConvertible.ToBoolean

19 [JScript] function IConvertible.ToBoolean(provider : IFormatProvider) : Boolean;

20 IConvertible.ToByte

22 [C#] byte IConvertible.ToByte(IFormatProvider provider);

23 [C++] unsigned char IConvertible::ToByte(IFormatProvider* provider);

24 [VB] Function ToByte(ByVal provider As IFormatProvider) As Byte Implements

1 IConvertible.ToByte

2 [JScript] function IConvertible.ToByte(provider : IFormatProvider) : Byte;

3 IConvertible.ToChar

5 [C#] char IConvertible.ToChar(IFormatProvider provider);

6 [C++] __wchar_t IConvertible::ToChar(IFormatProvider* provider);

7 [VB] Function ToChar(ByVal provider As IFormatProvider) As Char Implements

8 IConvertible.ToChar

9 [JScript] function IConvertible.ToChar(provider : IFormatProvider) : Char;

10 IConvertible.ToDateTime

12 [C#] DateTime IConvertible.ToDateTime(IFormatProvider provider);

13 [C++] DateTime IConvertible::ToDateTime(IFormatProvider* provider);

14 [VB] Function ToDateTime(ByVal provider As IFormatProvider) As DateTime

15 Implements IConvertible.ToDateTime

16 [JScript] function IConvertible.ToDateTime(provider : IFormatProvider) :

17 DateTime;

18 IConvertible.ToDecimal

20 [C#] decimal IConvertible.ToDecimal(IFormatProvider provider);

21 [C++] Decimal IConvertible::ToDecimal(IFormatProvider* provider);

22 [VB] Function ToDecimal(ByVal provider As IFormatProvider) As Decimal

23 Implements IConvertible.ToDecimal

24 [JScript] function IConvertible.ToDecimal(provider : IFormatProvider) : Decimal;

25 IConvertible.ToDouble

```

1
2 [C#] double IConvertible.ToDouble(IFormatProvider provider);
3 [C++] double IConvertible::ToDouble(IFormatProvider* provider);
4 [VB] Function ToDouble(ByVal provider As IFormatProvider) As Double
5 Implements IConvertible.ToDouble
6 [JScript] function IConvertible.ToDouble(provider : IFormatProvider) : double;
7     IConvertible.ToInt16
8
9 [C#] short IConvertible.ToInt16(IFormatProvider provider);
10 [C++] short IConvertible::ToInt16(IFormatProvider* provider);
11 [VB] Function ToInt16(ByVal provider As IFormatProvider) As Short
12 Implements IConvertible.ToInt16
13 [JScript] function IConvertible.ToInt16(provider : IFormatProvider) : Int16;
14     IConvertible.ToInt32
15
16 [C#] int IConvertible.ToInt32(IFormatProvider provider);
17 [C++] int IConvertible::ToInt32(IFormatProvider* provider);
18 [VB] Function ToInt32(ByVal provider As IFormatProvider) As Integer
19 Implements IConvertible.ToInt32
20 [JScript] function IConvertible.ToInt32(provider : IFormatProvider) : int;
21     IConvertible.ToInt64
22
23 [C#] long IConvertible.ToInt64(IFormatProvider provider);
24 [C++] __int64 IConvertible::ToInt64(IFormatProvider* provider);
25 [VB] Function ToInt64(ByVal provider As IFormatProvider) As Long Implements

```

```

1  IConvertible.ToInt64
2  [JScript] function IConvertible.ToInt64(provider : IFormatProvider) : long;
3
4  IConvertible.ToSByte
5
6  [C#] sbyte IConvertible.ToSByte(IFormatProvider provider);
7  [C++] char IConvertible::ToSByte(IFormatProvider* provider);
8  [VB] Function ToSByte(ByVal provider As IFormatProvider) As SByte
9  Implements IConvertible.ToSByte
10 [JScript] function IConvertible.ToSByte(provider : IFormatProvider) : SByte;
11
12 IConvertible.ToSingle
13
14 [C#] float IConvertible.ToSingle(IFormatProvider provider);
15 [C++] float IConvertible::ToSingle(IFormatProvider* provider);
16 [VB] Function ToSingle(ByVal provider As IFormatProvider) As Single
17 Implements IConvertible.ToSingle
18 [JScript] function IConvertible.ToSingle(provider : IFormatProvider) : float;
19
20 IConvertible.ToType
21
22 [C#] object IConvertible.ToType(Type type, IFormatProvider provider);
23 [C++] Object* IConvertible::ToType(Type* type, IFormatProvider* provider);
24 [VB] Function ToType(ByVal type As Type, ByVal provider As IFormatProvider)
25 As Object Implements IConvertible.ToType
26 [JScript] function IConvertible.ToType(type : Type, provider : IFormatProvider) :
27 Object;
28
29 IConvertible.ToUInt16

```

```

1
2 [C#] ushort IConvertible.ToUInt16(IFormatProvider provider);
3 [C++] unsigned short IConvertible::ToUInt16(IFormatProvider* provider);
4 [VB] Function ToUInt16(ByVal provider As IFormatProvider) As UInt16
5 Implements IConvertible.ToUInt16
6 [JScript] function IConvertible.ToUInt16(provider : IFormatProvider) : UInt16;
7     IConvertible.ToUInt32
8
9 [C#] uint IConvertible.ToUInt32(IFormatProvider provider);
10 [C++] unsigned int IConvertible::ToUInt32(IFormatProvider* provider);
11 [VB] Function ToUInt32(ByVal provider As IFormatProvider) As UInt32
12 Implements IConvertible.ToUInt32
13 [JScript] function IConvertible.ToUInt32(provider : IFormatProvider) : UInt32;
14     IConvertible.ToUInt64
15
16 [C#] ulong IConvertible.ToUInt64(IFormatProvider provider);
17 [C++] unsigned __int64 IConvertible::ToUInt64(IFormatProvider* provider);
18 [VB] Function ToUInt64(ByVal provider As IFormatProvider) As UInt64
19 Implements IConvertible.ToUInt64
20 [JScript] function IConvertible.ToUInt64(provider : IFormatProvider) : UInt64;
21     ToString
22
23 [C#] public override string ToString();
24 [C++] public: String* ToString();
25 [VB] Overrides Public Function ToString() As String

```

1 [JScript] public override function ToString() : String; Converts the numeric value
2 of this instance to its equivalent **String** representation.

3
4 *Description*

5 Converts the numeric value of this instance to its equivalent **String**
6 representation.

7 *Return Value:* The **System.String** representation of the value of this instance,
8 consisting of a minus sign if the value is negative, and a sequence of digits ranging
9 from 0 to 9 with no leading zeroes.

10 The return value is formatted with the general format specifier ("G") and
11 the **System.Globalization.NumberFormatInfo** for the current culture.

12 **ToString**

13
14 [C#] public string ToString(IFormatProvider provider);

15 [C++] public: __sealed String* ToString(IFormatProvider* provider);

16 [VB] NotOverridable Public Function ToString(ByVal provider As
17 IFormatProvider) As String

18 [JScript] public function ToString(provider : IFormatProvider) : String;

19
20 *Description*

21 Converts the numeric value of this instance to its equivalent **String**
22 representation using the specified culture-specific format information.

23 *Return Value:* The **System.String** representation of the value of this instance as
24 specified by *provider* .

This instance is formatted with the general format specifier ("G"). An **System.IFormatProvider** interface implementation which supplies culture-specific formatting information.

ToString

[C#] public string ToString(string format);

[C++] public: String* ToString(String* format);

[VB] Public Function ToString(ByVal format As String) As String

[JScript] public function ToString(format : String) : String;

Description

Converts the numeric value of this instance to its equivalent **String** representation, using the specified format.

Return Value: The **System.String** representation of the value of this instance as specified by *format* .

If *format* is **null** or an empty string, the return value of this instance is formatted with the general format specifier ("G"). A format string.

ToString

[C#] public string ToString(string format, IFormatProvider provider);

[C++] public: __sealed String* ToString(String* format, IFormatProvider* provider);

[VB] NotOverridable Public Function ToString(ByVal format As String, ByVal provider As IFormatProvider) As String

[JScript] public function ToString(format : String, provider : IFormatProvider) :

String;

Description

Converts the numeric value of this instance to its equivalent **String** representation using the specified format and culture-specific format information.

Return Value: The **System.String** representation of the value of this instance as specified by *format* and *provider* .

If *format* is **null** or an empty string, the return value for this instance is formatted with the general format specifier ("G"). A format specification. An **System.IFormatProvider** interface implementation which supplies culture-specific formatting information about this instance.

Int32 structure (System)

ToString

Description

Represents a 32-bit signed integer.

The **Int32** value type represents signed integers with values ranging from negative 2,147,483,648 through positive 2,147,483,647.

ToString

[C#] public const int MaxValue;

[C++] public: const int MaxValue;

[VB] Public Const MaxValue As Integer

[JScript] public var MaxValue : int;

1
2 *Description*

3 A constant representing the largest possible value of **Int32** .

4 The value of this constant is 2,147,483,647; that is, hexadecimal
5 0x7FFFFFFF.

6 **ToString**

7
8 [C#] public const int MinValue;

9 [C++] public: const int MinValue;

10 [VB] Public Const MinValue As Integer

11 [JScript] public var MinValue : int;

12
13 *Description*

14 A constant representing the smallest possible value of **Int32** .

15 The value of this constant is -2,147,483,648; that is, hexadecimal
16 0x80000000.

17 **CompareTo**

18
19 [C#] public int CompareTo(object value);

20 [C++] public: __sealed int CompareTo(Object* value);

21 [VB] NotOverridable Public Function CompareTo(ByVal value As Object) As
22 Integer

23 [JScript] public function CompareTo(value : Object) : int;

24
25 *Description*

Compares this instance to a specified object and returns an indication of their relative values.

Return Value: A signed number indicating the relative values of this instance and value .

Any instance of **Int32** , regardless of its value, is considered greater than **null** . An object to compare, or **null**.

Equals

[C#] public override bool Equals(object obj);

[C++] public: bool Equals(Object* obj);

[VB] Overrides Public Function Equals(ByVal obj As Object) As Boolean

[JScript] public override function Equals(obj : Object) : Boolean;

Description

Returns a value indicating whether this instance is equal to a specified object.

Return Value: **true** if *obj* is an instance of **Int32** and equals the value of this instance; otherwise, **false** . An object to compare with this instance.

GetHashCode

[C#] public override int GetHashCode();

[C++] public: int GetHashCode();

[VB] Overrides Public Function GetHashCode() As Integer

[JScript] public override function GetHashCode() : int;

1
2 *Description*

3 Returns the hash code for this instance.

4 *Return Value:* A 32-bit signed integer hash code.

5 **GetTypeCode**

6
7 [C#] public TypeCode GetTypeCode();

8 [C++] public: __sealed TypeCode GetTypeCode();

9 [VB] NotOverridable Public Function GetTypeCode() As TypeCode

10 [JScript] public function GetTypeCode() : TypeCode;

11
12 *Description*

13 Returns the **TypeCode** for value type **Int32** .

14 *Return Value:* The enumerated constant, **System.TypeCode.Int32** .

15 **Parse**

16
17 [C#] public static int Parse(string s);

18 [C++] public: static int Parse(String* s);

19 [VB] Public Shared Function Parse(ByVal s As String) As Integer

20 [JScript] public static function Parse(s : String) : int; Converts the **String**
21 representation of a number to its 32-bit signed integer equivalent.

22
23 *Description*

Converts the **String** representation of a number to its 32-bit signed integer equivalent.

Return Value: An 32-bit signed integer equivalent to the number contained in *s* .

s contains a number of the form: [ws][sign]digits[ws] Items in square brackets ('[' and ']') are optional, and other items are as follows. A **System.String** containing a number to convert.

Parse

[C#] public static int Parse(string s, IFormatProvider provider);

[C++] public: static int Parse(String* s, IFormatProvider* provider);

[VB] Public Shared Function Parse(ByVal s As String, ByVal provider As IFormatProvider) As Integer

[JScript] public static function Parse(s : String, provider : IFormatProvider) : int;

Description

Converts the **String** representation of a number in a specified culture-specific format to its 32-bit signed integer equivalent.

Return Value: A 32-bit signed integer equivalent to the number specified in *s* .

s contains a number of the form: [ws][sign]digits[ws] Items in square brackets ('[' and ']') are optional, and other items are as follows. A **System.String** containing a number to convert. An **System.IFormatProvider** interface implementation which supplies culture-specific formatting information about *s*.

Parse

[C#] public static int Parse(string s, NumberStyles style);

1 [C++] public: static int Parse(String* s, NumberStyles style);

2 [VB] Public Shared Function Parse(ByVal s As String, ByVal style As
3 NumberStyles) As Integer

4 [JScript] public static function Parse(s : String, style : NumberStyles) : int;

6 *Description*

7 Converts the **String** representation of a number in a specified style to its
8 32-bit signed integer equivalent.

9 *Return Value:* An 32-bit signed integer equivalent to the number specified in *s* .

10 *s* contains a number of the form: [ws][sign]digits[ws] Items in square
11 brackets ('[' and ']') are optional, and other items are as follows. A **System.String**
12 containing a number to convert. The combination of one or more
13 **System.Globalization.NumberStyles** constants that indicate the permitted format
14 of *s*.

15 *Parse*

16
17 [C#] public static int Parse(string s, NumberStyles style, IFormatProvider
18 provider);

19 [C++] public: static int Parse(String* s, NumberStyles style, IFormatProvider*
20 provider);

21 [VB] Public Shared Function Parse(ByVal s As String, ByVal style As
22 NumberStyles, ByVal provider As IFormatProvider) As Integer

23 [JScript] public static function Parse(s : String, style : NumberStyles, provider :
24 IFormatProvider) : int;

1
2 *Description*

3 Converts the **String** representation of a number in a specified style and
4 culture-specific format to its 32-bit signed integer equivalent.

5 *Return Value:* An 32-bit signed integer equivalent to the number specified in *s* .

6 *s* contains a number of the form: [ws][sign]digits[ws] Items in square
7 brackets ('[' and ']') are optional, and other items are as follows. A **System.String**
8 containing a number to convert. The combination of one or more
9 **System.Globalization.NumberStyles** constants that indicate the permitted format
10 of *s*. An **System.IFormatProvider** interface implementation which supplies
11 culture-specific formatting information about *s*.

12 **Convertible.ToBoolean**

13
14 [C#] bool Convertible.ToBoolean(IFormatProvider provider);

15 [C++] bool Convertible::ToBoolean(IFormatProvider* provider);

16 [VB] Function ToBoolean(ByVal provider As IFormatProvider) As Boolean

17 Implements Convertible.ToBoolean

18 [JScript] function Convertible.ToBoolean(provider : IFormatProvider) : Boolean;

19 **Convertible.ToByte**

20
21 [C#] byte Convertible.ToByte(IFormatProvider provider);

22 [C++] unsigned char Convertible::ToByte(IFormatProvider* provider);

23 [VB] Function ToByte(ByVal provider As IFormatProvider) As Byte Implements

24 Convertible.ToByte

25 [JScript] function Convertible.ToByte(provider : IFormatProvider) : Byte;

IConvertible.ToChar

[C#] char IConvertible.ToChar(IFormatProvider provider);

[C++] __wchar_t IConvertible::ToChar(IFormatProvider* provider);

[VB] Function ToChar(ByVal provider As IFormatProvider) As Char Implements

IConvertible.ToChar

[JScript] function IConvertible.ToChar(provider : IFormatProvider) : Char;

IConvertible.ToDateTime

[C#] DateTime IConvertible.ToDateTime(IFormatProvider provider);

[C++] DateTime IConvertible::ToDateTime(IFormatProvider* provider);

[VB] Function ToDateTime(ByVal provider As IFormatProvider) As DateTime

Implements IConvertible.ToDateTime

[JScript] function IConvertible.ToDateTime(provider : IFormatProvider) :

DateTime;

IConvertible.ToDecimal

[C#] decimal IConvertible.ToDecimal(IFormatProvider provider);

[C++] Decimal IConvertible::ToDecimal(IFormatProvider* provider);

[VB] Function ToDecimal(ByVal provider As IFormatProvider) As Decimal

Implements IConvertible.ToDecimal

[JScript] function IConvertible.ToDecimal(provider : IFormatProvider) : Decimal;

IConvertible.ToDouble

[C#] double IConvertible.ToDouble(IFormatProvider provider);

```

1  [C++] double IConvertible::ToDouble(IFormatProvider* provider);
2  [VB] Function ToDouble(ByVal provider As IFormatProvider) As Double
3  Implements IConvertible.ToDouble
4  [JScript] function IConvertible.ToDouble(provider : IFormatProvider) : double;
5      IConvertible.ToInt16
6
7  [C#] short IConvertible.ToInt16(IFormatProvider provider);
8  [C++] short IConvertible::ToInt16(IFormatProvider* provider);
9  [VB] Function ToInt16(ByVal provider As IFormatProvider) As Short
10 Implements IConvertible.ToInt16
11 [JScript] function IConvertible.ToInt16(provider : IFormatProvider) : Int16;
12     IConvertible.ToInt32
13
14 [C#] int IConvertible.ToInt32(IFormatProvider provider);
15 [C++] int IConvertible::ToInt32(IFormatProvider* provider);
16 [VB] Function ToInt32(ByVal provider As IFormatProvider) As Integer
17 Implements IConvertible.ToInt32
18 [JScript] function IConvertible.ToInt32(provider : IFormatProvider) : int;
19     IConvertible.ToInt64
20
21 [C#] long IConvertible.ToInt64(IFormatProvider provider);
22 [C++] __int64 IConvertible::ToInt64(IFormatProvider* provider);
23 [VB] Function ToInt64(ByVal provider As IFormatProvider) As Long Implements
24 IConvertible.ToInt64
25 [JScript] function IConvertible.ToInt64(provider : IFormatProvider) : long;

```


1 IConvertible.ToSByte

2
3 [C#] sbyte IConvertible.ToSByte(IFormatProvider provider);

4 [C++] char IConvertible::ToSByte(IFormatProvider* provider);

5 [VB] Function ToSByte(ByVal provider As IFormatProvider) As SByte

6 Implements IConvertible.ToSByte

7 [JScript] function IConvertible.ToSByte(provider : IFormatProvider) : SByte;

8 IConvertible.ToSingle

9
10 [C#] float IConvertible.ToSingle(IFormatProvider provider);

11 [C++] float IConvertible::ToSingle(IFormatProvider* provider);

12 [VB] Function ToSingle(ByVal provider As IFormatProvider) As Single

13 Implements IConvertible.ToSingle

14 [JScript] function IConvertible.ToSingle(provider : IFormatProvider) : float;

15 IConvertible.ToType

16
17 [C#] object IConvertible.ToType(Type type, IFormatProvider provider);

18 [C++] Object* IConvertible::ToType(Type* type, IFormatProvider* provider);

19 [VB] Function ToType(ByVal type As Type, ByVal provider As IFormatProvider)

20 As Object Implements IConvertible.ToType

21 [JScript] function IConvertible.ToType(type : Type, provider : IFormatProvider) :

22 Object;

23 IConvertible.ToUInt16

24
25 [C#] ushort IConvertible.ToUInt16(IFormatProvider provider);

```

1  [C++] unsigned short IConvertible::ToUInt16(IFormatProvider* provider);
2  [VB] Function ToUInt16(ByVal provider As IFormatProvider) As UInt16
3  Implements IConvertible.ToUInt16
4  [JScript] function IConvertible.ToUInt16(provider : IFormatProvider) : UInt16;
5      IConvertible.ToUInt32
6
7  [C#] uint IConvertible.ToUInt32(IFormatProvider provider);
8  [C++] unsigned int IConvertible::ToUInt32(IFormatProvider* provider);
9  [VB] Function ToUInt32(ByVal provider As IFormatProvider) As UInt32
10 Implements IConvertible.ToUInt32
11 [JScript] function IConvertible.ToUInt32(provider : IFormatProvider) : UInt32;
12     IConvertible.ToUInt64
13
14 [C#] ulong IConvertible.ToUInt64(IFormatProvider provider);
15 [C++] unsigned __int64 IConvertible::ToUInt64(IFormatProvider* provider);
16 [VB] Function ToUInt64(ByVal provider As IFormatProvider) As UInt64
17 Implements IConvertible.ToUInt64
18 [JScript] function IConvertible.ToUInt64(provider : IFormatProvider) : UInt64;
19     ToString
20
21 [C#] public override string ToString();
22 [C++] public: String* ToString();
23 [VB] Overrides Public Function ToString() As String
24 [JScript] public override function ToString() : String; Converts the numeric value
25 of this instance to its equivalent String representation.

```

1
2 *Description*

3 Converts the numeric value of this instance to its equivalent **String**
4 representation.

5 *Return Value:* The **System.String** representation of the value of this instance,
6 consisting of a negative sign if the value is negative, and a sequence of digits
7 ranging from 0 to 9 with no leading zeroes.

8 The return value is formatted with the general format specifier ("G") and
9 the **System.Globalization.NumberFormatInfo** for the current culture.

10 **ToString**

11
12 [C#] public string ToString(IFormatProvider provider);
13 [C++] public: __sealed String* ToString(IFormatProvider* provider);
14 [VB] NotOverridable Public Function ToString(ByVal provider As
15 IFormatProvider) As String
16 [JScript] public function ToString(provider : IFormatProvider) : String;

17
18 *Description*

19 Converts the numeric value of this instance to its equivalent **String**
20 representation using the specified culture-specific format information.

21 *Return Value:* The **System.String** representation of the value of this instance as
22 specified by *provider* .

23 This instance is formatted with the general format specifier ("G"). An
24 **System.IFormatProvider** interface implementation which supplies culture-
25 specific formatting information.

ToString

```
[C#] public string ToString(string format);  
[C++] public: String* ToString(String* format);  
[VB] Public Function ToString(ByVal format As String) As String  
[JScript] public function ToString(format : String) : String;
```

Description

Converts the numeric value of this instance to its equivalent **String** representation, using the specified format.

Return Value: The **System.String** representation of the value of this instance as specified by *format* .

If *format* is **null** or an empty string (""), the return value of this instance is formatted with the general format specifier ("G"). A format string.

ToString

```
[C#] public string ToString(string format, IFormatProvider provider);  
[C++] public: __sealed String* ToString(String* format, IFormatProvider*  
provider);  
[VB] NotOverridable Public Function ToString(ByVal format As String, ByVal  
provider As IFormatProvider) As String  
[JScript] public function ToString(format : String, provider : IFormatProvider) :  
String;
```

Description

Converts the numeric value of this instance to its equivalent **String** representation using the specified format and culture-specific format information.

Return Value: The **System.String** representation of the value of this instance as specified by *format* and *provider* .

If *format* is **null** or an empty string (""), the return value for this instance is formatted with the general format specifier ("G"). A format specification. An **System.IFormatProvider** interface implementation which supplies culture-specific formatting information.

Int64 structure (System)

ToString

Description

Represents a 64-bit signed integer.

The **Int64** value type represents integers with values ranging from negative 9,223,372,036,854,775,808 through positive 9,223,372,036,854,775,807.

ToString

[C#] public const long MaxValue;

[C++] public: const __int64 MaxValue;

[VB] Public Const MaxValue As Long

[JScript] public var MaxValue : long;

Description

A constant representing the largest possible value of **Int64** .

The value of this constant is 9,223,372,036,854,775,807; that is,
hexadecimal 0x7FFFFFFFFFFFFFFF.

ToString

[C#] public const long MinValue;

[C++] public: const __int64 MinValue;

[VB] Public Const MinValue As Long

[JScript] public var MinValue : long;

Description

A constant representing the smallest possible value of **Int64** .

The value of this constant is negative 9,223,372,036,854,775,808; that is,
hexadecimal 0x8000000000000000.

CompareTo

[C#] public int CompareTo(object value);

[C++] public: __sealed int CompareTo(Object* value);

[VB] NotOverridable Public Function CompareTo(ByVal value As Object) As

Integer

[JScript] public function CompareTo(value : Object) : int;

Description

Compares this instance to a specified object and returns an indication of
their relative values.

1 *Return Value:* A signed number indicating the relative values of this instance and
2 *value* .

3 An **Int64** , regardless of its value, is considered greater than a null
4 reference. An object to compare, or **null**.

5 Equals

6
7 [C#] public override bool Equals(object obj);

8 [C++] public: bool Equals(Object* obj);

9 [VB] Overrides Public Function Equals(ByVal obj As Object) As Boolean

10 [JScript] public override function Equals(obj : Object) : Boolean;

11 12 *Description*

13 Returns a value indicating whether this instance is equal to a specified
14 object.

15 *Return Value:* **true** if *obj* is an instance of **Int64** and equals the value of this
16 instance; otherwise, **false** . An object to compare with this instance.

17 GetHashCode

18
19 [C#] public override int GetHashCode();

20 [C++] public: int GetHashCode();

21 [VB] Overrides Public Function GetHashCode() As Integer

22 [JScript] public override function GetHashCode() : int;

23 24 *Description*

1 Returns the hash code for this instance.

2 *Return Value:* A 32-bit signed integer hash code.

3 GetTypeCode

4
5 [C#] public TypeCode GetTypeCode();

6 [C++] public: __sealed TypeCode GetTypeCode();

7 [VB] NotOverridable Public Function GetTypeCode() As TypeCode

8 [JScript] public function GetTypeCode() : TypeCode;

9
10 *Description*

11 Returns the **TypeCode** for value type **Int64** .

12 *Return Value:* The enumerated constant, **System.TypeCode.Int64** .

13 Parse

14
15 [C#] public static long Parse(string s);

16 [C++] public: static __int64 Parse(String* s);

17 [VB] Public Shared Function Parse(ByVal s As String) As Long

18 [JScript] public static function Parse(s : String) : long; Converts the **String**

19 representation of a number to its 64-bit signed integer equivalent.

20
21 *Description*

22 Converts the **String** representation of a number to its 64-bit signed integer
23 equivalent.

24 *Return Value:* A 64-bit signed integer equivalent to the number contained in *s* .

1 *s* contains a number of the form: [ws][sign]digits[ws] Items in square
2 brackets ('[' and ']') are optional, and other items are as follows. A **System.String**
3 containing a number to convert.

4 Parse

5
6 [C#] public static long Parse(string s, IFormatProvider provider);
7 [C++] public: static __int64 Parse(String* s, IFormatProvider* provider);
8 [VB] Public Shared Function Parse(ByVal s As String, ByVal provider As
9 IFormatProvider) As Long
10 [JScript] public static function Parse(s : String, provider : IFormatProvider) : long;

11 *Description*

12 Converts the **String** representation of a number in a specified culture-
13 specific format to its 64-bit signed integer equivalent.

14 *Return Value:* A 64-bit signed integer equivalent to the number specified in *s* .

15 *s* contains a number of the form: [ws][sign]digits[ws] Items in square
16 brackets ('[' and ']') are optional, and other items are as follows. A **System.String**
17 containing a number to convert. An **System.IFormatProvider** interface
18 implementation which supplies culture-specific formatting information about *s*.
19

20 Parse

21
22 [C#] public static long Parse(string s, NumberStyles style);
23 [C++] public: static __int64 Parse(String* s, NumberStyles style);
24 [VB] Public Shared Function Parse(ByVal s As String, ByVal style As
25 NumberStyles) As Long

1 [JScript] public static function Parse(s : String, style : NumberStyles) : long;

3 *Description*

4 Converts the **String** representation of a number in a specified style to its
5 64-bit signed integer equivalent.

6 *Return Value:* A 64-bit signed integer equivalent to the number specified in *s* .

7 *s* contains a number of the form: [ws][sign]digits[ws] Items in square
8 brackets ('[' and ']') are optional, and other items are as follows. A **System.String**
9 containing a number to convert. The combination of one or more
10 **System.Globalization.NumberStyles** constants that indicate the permitted format
11 of *s*.

12 Parse

13
14 [C#] public static long Parse(string s, NumberStyles style, IFormatProvider
15 provider);

16 [C++] public: static __int64 Parse(String* s, NumberStyles style,
17 IFormatProvider* provider);

18 [VB] Public Shared Function Parse(ByVal s As String, ByVal style As
19 NumberStyles, ByVal provider As IFormatProvider) As Long

20 [JScript] public static function Parse(s : String, style : NumberStyles, provider :
21 IFormatProvider) : long;

22
23 *Description*

Converts the **String** representation of a number in a specified style and culture-specific format to its 64-bit signed integer equivalent.

Return Value: A 64-bit signed integer equivalent to the number specified in *s*.

s contains a number of the form: [ws][sign]digits[ws] Items in square brackets ('[' and ']') are optional, and other items are as follows. A **System.String** containing a number to convert. The combination of one or more **System.Globalization.NumberStyles** constants that indicate the permitted format of *s*. An **System.IFormatProvider** interface implementation which supplies culture-specific formatting information about *s*.

ICconvertible.ToBoolean

[C#] bool **ICconvertible.ToBoolean**(IFormatProvider provider);

[C++] bool **ICconvertible::ToBoolean**(IFormatProvider* provider);

[VB] Function **ToBoolean**(ByVal provider As IFormatProvider) As Boolean

Implements **ICconvertible.ToBoolean**

[JScript] function **ICconvertible.ToBoolean**(provider : IFormatProvider) : Boolean;

ICconvertible.ToByte

[C#] byte **ICconvertible.ToByte**(IFormatProvider provider);

[C++] unsigned char **ICconvertible::ToByte**(IFormatProvider* provider);

[VB] Function **ToByte**(ByVal provider As IFormatProvider) As Byte Implements

ICconvertible.ToByte

[JScript] function **ICconvertible.ToByte**(provider : IFormatProvider) : Byte;

ICconvertible.ToChar

```

1
2 [C#] char IConvertible.ToChar(IFormatProvider provider);
3 [C++] __wchar_t IConvertible::ToChar(IFormatProvider* provider);
4 [VB] Function ToChar(ByVal provider As IFormatProvider) As Char Implements
5 IConvertible.ToChar
6 [JScript] function IConvertible.ToChar(provider : IFormatProvider) : Char;
7     IConvertible.ToDateTime
8
9 [C#] DateTime IConvertible.ToDateTime(IFormatProvider provider);
10 [C++] DateTime IConvertible::ToDateTime(IFormatProvider* provider);
11 [VB] Function ToDateTime(ByVal provider As IFormatProvider) As DateTime
12 Implements IConvertible.ToDateTime
13 [JScript] function IConvertible.ToDateTime(provider : IFormatProvider) :
14 DateTime;
15     IConvertible.ToDecimal
16
17 [C#] decimal IConvertible.ToDecimal(IFormatProvider provider);
18 [C++] Decimal IConvertible::ToDecimal(IFormatProvider* provider);
19 [VB] Function ToDecimal(ByVal provider As IFormatProvider) As Decimal
20 Implements IConvertible.ToDecimal
21 [JScript] function IConvertible.ToDecimal(provider : IFormatProvider) : Decimal;
22     IConvertible.ToDouble
23
24 [C#] double IConvertible.ToDouble(IFormatProvider provider);
25 [C++] double IConvertible::ToDouble(IFormatProvider* provider);

```

```

1  [VB] Function ToDouble(ByVal provider As IFormatProvider) As Double
2  Implements IConvertible.ToDouble
3  [JScript] function IConvertible.ToDouble(provider : IFormatProvider) : double;
4      IConvertible.ToInt16
5
6  [C#] short IConvertible.ToInt16(IFormatProvider provider);
7  [C++] short IConvertible::ToInt16(IFormatProvider* provider);
8  [VB] Function ToInt16(ByVal provider As IFormatProvider) As Short
9  Implements IConvertible.ToInt16
10 [JScript] function IConvertible.ToInt16(provider : IFormatProvider) : Int16;
11     IConvertible.ToInt32
12
13 [C#] int IConvertible.ToInt32(IFormatProvider provider);
14 [C++] int IConvertible::ToInt32(IFormatProvider* provider);
15 [VB] Function ToInt32(ByVal provider As IFormatProvider) As Integer
16 Implements IConvertible.ToInt32
17 [JScript] function IConvertible.ToInt32(provider : IFormatProvider) : int;
18     IConvertible.ToInt64
19
20 [C#] long IConvertible.ToInt64(IFormatProvider provider);
21 [C++] __int64 IConvertible::ToInt64(IFormatProvider* provider);
22 [VB] Function ToInt64(ByVal provider As IFormatProvider) As Long Implements
23 IConvertible.ToInt64
24 [JScript] function IConvertible.ToInt64(provider : IFormatProvider) : long;
25     IConvertible.ToSByte
    
```

```

1
2 [C#] sbyte IConvertible.ToSByte(IFormatProvider provider);
3 [C++] char IConvertible::ToSByte(IFormatProvider* provider);
4 [VB] Function ToSByte(ByVal provider As IFormatProvider) As SByte
5 Implements IConvertible.ToSByte
6 [JScript] function IConvertible.ToSByte(provider : IFormatProvider) : SByte;
7     IConvertible.ToSingle
8
9 [C#] float IConvertible.ToSingle(IFormatProvider provider);
10 [C++] float IConvertible::ToSingle(IFormatProvider* provider);
11 [VB] Function ToSingle(ByVal provider As IFormatProvider) As Single
12 Implements IConvertible.ToSingle
13 [JScript] function IConvertible.ToSingle(provider : IFormatProvider) : float;
14     IConvertible.ToType
15
16 [C#] object IConvertible.ToType(Type type, IFormatProvider provider);
17 [C++] Object* IConvertible::ToType(Type* type, IFormatProvider* provider);
18 [VB] Function ToType(ByVal type As Type, ByVal provider As IFormatProvider)
19 As Object Implements IConvertible.ToType
20 [JScript] function IConvertible.ToType(type : Type, provider : IFormatProvider) :
21 Object;
22     IConvertible.ToUInt16
23
24 [C#] ushort IConvertible.ToUInt16(IFormatProvider provider);
25 [C++] unsigned short IConvertible::ToUInt16(IFormatProvider* provider);

```

```

1  [VB] Function ToUInt16(ByVal provider As IFormatProvider) As UInt16
2  Implements IConvertible.ToUInt16
3  [JScript] function IConvertible.ToUInt16(provider : IFormatProvider) : UInt16;
4      IConvertible.ToUInt32
5
6  [C#] uint IConvertible.ToUInt32(IFormatProvider provider);
7  [C++] unsigned int IConvertible::ToUInt32(IFormatProvider* provider);
8  [VB] Function ToUInt32(ByVal provider As IFormatProvider) As UInt32
9  Implements IConvertible.ToUInt32
10 [JScript] function IConvertible.ToUInt32(provider : IFormatProvider) : UInt32;
11     IConvertible.ToUInt64
12
13 [C#] ulong IConvertible.ToUInt64(IFormatProvider provider);
14 [C++] unsigned __int64 IConvertible::ToUInt64(IFormatProvider* provider);
15 [VB] Function ToUInt64(ByVal provider As IFormatProvider) As UInt64
16 Implements IConvertible.ToUInt64
17 [JScript] function IConvertible.ToUInt64(provider : IFormatProvider) : UInt64;
18     ToString
19
20 [C#] public override string ToString();
21 [C++] public: String* ToString();
22 [VB] Overrides Public Function ToString() As String
23 [JScript] public override function ToString() : String; Converts the numeric value
24 of this instance to its equivalent String representation.
25

```

Description

Converts the numeric value of this instance to its equivalent **String** representation.

Return Value: The **System.String** representation of the value of this instance, consisting of a minus sign if the value is negative, and a sequence of digits ranging from 0 to 9 with no leading zeroes.

The return value is formatted with the general format specifier ("G") and the **System.Globalization.NumberFormatInfo** for the current culture.

ToString

[C#] public string ToString(IFormatProvider provider);

[C++] public: __sealed String* ToString(IFormatProvider* provider);

[VB] NotOverridable Public Function ToString(ByVal provider As IFormatProvider) As String

[JScript] public function ToString(provider : IFormatProvider) : String;

Description

Converts the numeric value of this instance to its equivalent **String** representation using the specified culture-specific format information.

Return Value: The **System.String** representation of the value of this instance as specified by *provider*.

This instance is formatted with the general format specifier ("G"). An **System.IFormatProvider** interface implementation which supplies culture-specific formatting information.

ToString

```
[C#] public string ToString(string format);  
[C++] public: String* ToString(String* format);  
[VB] Public Function ToString(ByVal format As String) As String  
[JScript] public function ToString(format : String) : String;
```

Description

Converts the numeric value of this instance to its equivalent **String** representation, using the specified format.

Return Value: The **System.String** representation of the value of this instance as specified by *format* .

If *format* is **null** or an empty string (""), the return value of this instance is formatted with the general format specifier ("G"). A format string.

ToString

```
[C#] public string ToString(string format, IFormatProvider provider);  
[C++] public: __sealed String* ToString(String* format, IFormatProvider*  
provider);  
[VB] NotOverridable Public Function ToString(ByVal format As String, ByVal  
provider As IFormatProvider) As String  
[JScript] public function ToString(format : String, provider : IFormatProvider) :  
String;
```

Description

Converts the numeric value of this instance to its equivalent **String** representation using the specified format and culture-specific format information.

Return Value: The **System.String** representation of the value of this instance as specified by *format* and *provider* .

If *format* is **null** or an empty string (""), the return value for this instance is formatted with the general format specifier ("G"). A format specification. An **System.IFormatProvider** interface implementation which supplies culture-specific formatting information about this instance.

IntPtr structure (System)

ToString

Description

A platform-specific type that is used to represent a pointer or a handle.

The **System.IntPtr** type is designed to be a platform-specific, machine-sized integer. That is, an instance of this type is expected to be 32-bits on 32-bit hardware and operating systems, and 64-bits on 64-bit hardware and operating systems.

ToString

[C#] public static readonly IntPtr Zero;

[C++] public: static IntPtr Zero;

[VB] Public Shared ReadOnly Zero As IntPtr

[JScript] public static var Zero : IntPtr;

1
2 *Description*

3 A read-only field that represents an uninitialized pointer or handle.

4 The value of this field is not equivalent to **null** , but is instead a pointer
5 which has not been assigned any value whatsoever. Use this field to efficiently
6 determine whether an instance of **IntPtr** has been set.

7 **IntPtr**

8 *Example Syntax:*

9 **ToString**

10
11 [C#] public IntPtr(int value);

12 [C++] public: IntPtr(int value);

13 [VB] Public Sub New(ByVal value As Integer)

14 [JScript] public function IntPtr(value : int); Initializes a new instance of the
15 **System.IntPtr** structure.

16
17 *Description*

18 Initializes a new instance of the **System.IntPtr** structure to the specified
19 32-bit pointer or handle. A pointer or handle contained in a 32-bit signed integer.

20 **IntPtr**

21 *Example Syntax:*

22 **ToString**

23
24 [C#] public IntPtr(long value);

25 [C++] public: IntPtr(__int64 value);

1 [VB] Public Sub New(ByVal value As Long)

2 [JScript] public function IntPtr(value : long);

3
4 *Description*

5 Initializes a new instance of the **System.IntPtr** structure to the specified
6 64-bit pointer.

7 An exception is only thrown if the value of *value* requires more bits than
8 the current platform supports. A pointer or handle contained in a 64-bit signed
9 integer.

10 IntPtr

11 *Example Syntax:*

12 ToString

13
14 [C#] unsafe public IntPtr(void* value);

15 [C++] public: IntPtr(void* value);

16 Size

17 ToString

18
19 [C#] public static int Size {get;}

20 [C++] public: __property static int get_Size();

21 [VB] Public Shared ReadOnly Property Size As Integer

22 [JScript] public static function get Size() : int;

23
24 *Description*

25 Gets the size of this instance.

Equals

[C#] public override bool Equals(object obj);

[C++] public: bool Equals(Object* obj);

[VB] Overrides Public Function Equals(ByVal obj As Object) As Boolean

[JScript] public override function Equals(obj : Object) : Boolean;

Description

Returns a value indicating whether this instance is equal to a specified object.

Return Value: **true** if *obj* is an instance of **IntPtr** and equals the value of this instance; otherwise, **false** . An object to compare with this instance or **null**.

GetHashCode

[C#] public override int GetHashCode();

[C++] public: int GetHashCode();

[VB] Overrides Public Function GetHashCode() As Integer

[JScript] public override function GetHashCode() : int;

Description

Returns the hash code for this instance.

Return Value: A 32-bit signed integer hash code.

op_Equality

[C#] public static bool operator ==(IntPtr value1, IntPtr value2);

1 [C++] public: static bool op_Equality(IntPtr value1, IntPtr value2);

2 [VB] returnValue = IntPtr.op_Equality(value1, value2)

3 [JScript] returnValue = value1 == value2;

4
5 *Description*

6 Determines whether two specified instances of **System.IntPtr** are equal.

7 *Return Value:* **true** if *value1* equals *value2* ; otherwise, **false** . An **IntPtr**. An
8 **IntPtr**.

9 op_Explicit

10
11 [C#] public static explicit operator IntPtr(int value);

12 [C++] public: static IntPtr op_Explicit(int value);

13 [VB] returnValue = IntPtr.op_Explicit(value)

14 [JScript] returnValue = IntPtr(value);

15
16 *Description*

17 Converts the value of a 32-bit signed integer to an **System.IntPtr** .

18 *Return Value:* A new instance of **System.IntPtr** initialized to *value* . A 32-bit
19 signed integer.

20 op_Explicit

21
22 [C#] public static explicit operator IntPtr(long value);

23 [C++] public: static IntPtr op_Explicit(__int64 value);

24 [VB] returnValue = IntPtr.op_Explicit(value)

25 [JScript] returnValue = IntPtr(value);

1
2 *Description*

3 Converts the value of a 64-bit signed integer to an **System.IntPtr** .

4 *Return Value:* A new instance of **System.IntPtr** initialized to *value* . A 64-bit
5 signed integer.

6 op_Explicit

7
8 [C#] public static explicit operator int(IntPtr value);

9 [C++] public: static int op_Explicit();

10 [VB] returnValue = IntPtr.op_Explicit(value)

11 [JScript] returnValue = Int32(value);

12
13 *Description*

14 Converts the value of the specified **System.IntPtr** instance to a 32-bit
15 signed integer.

16 An exception is only thrown if the value of *value* requires more bits than
17 the current platform supports. An **IntPtr**.

18 op_Explicit

19
20 [C#] unsafe public static explicit operator void*(IntPtr value);

21 [C++] public: static void* op_Explicit();

22 op_Explicit

23
24 [C#] public static explicit operator long(IntPtr value);

25 [C++] public: static __int64 op_Explicit();

1 [VB] returnValue = IntPtr.op_Explicit(value)

2 [JScript] returnValue = Int64(value);

3
4 *Description*

5 Converts the value of the specified **System.IntPtr** instance to a 64-bit
6 signed integer. An **IntPtr**.

7 op_Explicit

8
9 [C#] unsafe public static explicit operator IntPtr(void* value);

10 [C++] public: static IntPtr op_Explicit(void* value);

11 op_Inequality

12
13 [C#] public static bool operator !=(IntPtr value1, IntPtr value2);

14 [C++] public: static bool op_Inequality(IntPtr value1, IntPtr value2);

15 [VB] returnValue = IntPtr.op_Inequality(value1, value2)

16 [JScript] returnValue = value1 != value2;

17
18 *Description*

19 Determines whether two specified instances of **System.IntPtr** are not
20 equal.

21 *Return Value:* **true** if *value1* does not equal *value2* ; otherwise, **false** . An **IntPtr**.

22 An **IntPtr**.

23 ISerializable.GetObjectData

24
25 [C#] void ISerializable.GetObjectData(SerializationInfo info, StreamingContext


```

1 context);
2 [C++] void ISerializable::GetObjectData(SerializationInfo* info,
3 StreamingContext context);
4 [VB] Sub GetObjectData(ByVal info As SerializationInfo, ByVal context As
5 StreamingContext) Implements ISerializable.GetObjectData
6 [JScript] function ISerializable.GetObjectData(info : SerializationInfo, context :
7 StreamingContext);

```

ToInt32

```

9
10 [C#] public int ToInt32();
11 [C++] public: int ToInt32();
12 [VB] Public Function ToInt32() As Integer
13 [JScript] public function ToInt32() : int;

```

Description

Converts the value of this instance to a 32-bit signed integer.

Return Value: A 32-bit signed integer.

An exception is only thrown if the value of *value* requires more bits than the current platform supports.

ToInt64

```

21
22 [C#] public long ToInt64();
23 [C++] public: __int64 ToInt64();
24 [VB] Public Function ToInt64() As Long
25 [JScript] public function ToInt64() : long;

```

1
2 *Description*

3 Converts the value of this instance to a 64-bit signed integer.

4 *Return Value:* A a 64-bit signed integer.

5 ToPointer

6
7 [C#] unsafe public void* ToPointer();

8 [C++] public: void* ToPointer();

9
10 *Description*

11 Converts the value of this instance to a pointer to an unspecified type.

12 *Return Value:* A pointer to **System.Void** ; that is, a pointer to memory containing
13 data of an unspecified type.

14 ToString

15
16 [C#] public override string ToString();

17 [C++] public: String* ToString();

18 [VB] Overrides Public Function ToString() As String

19 [JScript] public override function ToString() : String;

20
21 *Description*

22 Converts the numeric value of this instance to its equivalent **String**
23 representation.

24 *Return Value:* The **System.String** representation of the value of this instance.

25 InvalidCastException class (System)

ToString

Description

The exception that is thrown for invalid casting or explicit conversion.

System.InvalidCastException is thrown if: For a conversion from a **System.Single** or a **System.Double** to a **System.Decimal** , the source value is infinity, Not-a-Number (NaN), or too large to be represented as the destination type.

InvalidCastException

Example Syntax:

ToString

[C#] public InvalidCastException();

[C++] public: InvalidCastException();

[VB] Public Sub New()

[JScript] public function InvalidCastException(); Initializes a new instance of the **System.InvalidCastException** class.

Description

Initializes a new instance of the **System.InvalidCastException** class with default properties.

The following table shows the initial property values for an instance of **System.InvalidCastException** .

InvalidCastException

Example Syntax:

ToString

[C#] public InvalidCastException(string message);

[C++] public: InvalidCastException(String* message);

[VB] Public Sub New(ByVal message As String)

[JScript] public function InvalidCastException(message : String);

Description

Initializes a new instance of the **System.InvalidCastException** class with a specified error message.

The following table shows the initial property values for an instance of **System.InvalidCastException**. The error message that explains the reason for the exception.

InvalidCastException

Example Syntax:

ToString

[C#] protected InvalidCastException(SerializationInfo info, StreamingContext context);

[C++] protected: InvalidCastException(SerializationInfo* info, StreamingContext context);

[VB] Protected Sub New(ByVal info As SerializationInfo, ByVal context As StreamingContext)

[JScript] protected function InvalidCastException(info : SerializationInfo, context

1 : StreamingContext);

2
3 *Description*

4 Initializes a new instance of the **System.InvalidCastException** class with
5 serialized data.

6 This constructor is called during deserialization to reconstitute the
7 exception object transmitted over a stream. For more information, see . The object
8 that holds the serialized object data. The contextual information about the source
9 or destination.

10 InvalidCastException

11 *Example Syntax:*

12 ToString

13
14 [C#] public InvalidCastException(string message, Exception innerException);

15 [C++] public: InvalidCastException(String* message, Exception*
16 innerException);

17 [VB] Public Sub New(ByVal message As String, ByVal innerException As
18 Exception)

19 [JScript] public function InvalidCastException(message : String, innerException :
20 Exception);

21
22 *Description*

23 Initializes a new instance of the **System.InvalidCastException** class with a
24 specified error message and a reference to the inner exception that is the root cause
25 of this exception.

When an **Exception** *X* is thrown as a direct result of a previous exception *Y*, the **System.Exception.InnerException** property of *X* should contain a reference to *Y*. The **InnerException** property returns the same value as was passed into the constructor, or **null** if the inner exception value was not supplied to the constructor. The error message that explains the reason for the exception. An instance of **System.Exception** that is the cause of the current **Exception**. If *innerException* is non-null, then the current **Exception** is raised in a catch block handling *innerException*.

HelpLink

HResult

InnerException

Message

Source

StackTrace

TargetSite

InvalidOperationException class (System)

ToString

Description

The exception that is thrown when a method call is invalid for the object's current state.

System.InvalidOperationException is used in cases when the failure to invoke a method is caused by reasons other than invalid arguments. For example, **System.InvalidOperationException** is thrown by:

System.Collections.IEnumerator.MoveNext if objects of a collection are modified after the enumerator is created.

InvalidOperationException

Example Syntax:

ToString

[C#] public InvalidOperationException();

[C++] public: InvalidOperationException();

[VB] Public Sub New()

[JScript] public function InvalidOperationException(); Initializes a new instance of the **System.InvalidOperationException** class.

Description

Initializes a new instance of the **System.InvalidOperationException** class with default properties.

The following table shows the initial property values for an instance of **System.InvalidOperationException**.

InvalidOperationException

Example Syntax:

ToString

[C#] public InvalidOperationException(string message);

[C++] public: InvalidOperationException(String* message);

[VB] Public Sub New(ByVal message As String)

[JScript] public function InvalidOperationException(message : String);

Description

Initializes a new instance of the **System.InvalidOperationException** class with a specified error message.

The following table shows the initial property values for an instance of **System.InvalidOperationException**. The error message that explains the reason for the exception.

InvalidOperationException

Example Syntax:

ToString

[C#] protected InvalidOperationException(SerializationInfo info, StreamingContext context);

[C++] protected: InvalidOperationException(SerializationInfo* info, StreamingContext context);

[VB] Protected Sub New(ByVal info As SerializationInfo, ByVal context As StreamingContext)

[JScript] protected function InvalidOperationException(info : SerializationInfo, context : StreamingContext);

Description

Initializes a new instance of the **System.InvalidOperationException** class with serialized data.

This constructor is called during deserialization to reconstitute the exception object transmitted over a stream. For more information, see . The object

that holds the serialized object data. The contextual information about the source or destination.

`InvalidOperationException`

Example Syntax:

`ToString`

[C#] `public InvalidOperationException(string message, Exception innerException);`

[C++] `public: InvalidOperationException(String* message, Exception* innerException);`

[VB] `Public Sub New(ByVal message As String, ByVal innerException As Exception)`

[JScript] `public function InvalidOperationException(message : String, innerException : Exception);`

Description

Initializes a new instance of the **System.InvalidOperationException** class with a specified error message and a reference to the inner exception that is the root cause of this exception.

When an **Exception** *X* is thrown as a direct result of a previous exception *Y*, the **System.Exception.InnerException** property of *X* should contain a reference to *Y*. The **InnerException** property returns the same value as was passed into the constructor, or **null** if the inner exception value was not supplied to the constructor. The error message that explains the reason for the exception. An instance of **System.Exception** that is the cause of the current **Exception**. If

1 *innerException* is non-null, then the current **Exception** is raised in a catch block
2 handling *innerException* .

3 HelpLink

4 HResult

5 InnerException

6 Message

7 Source

8 StackTrace

9 TargetSite

10 InvalidProgramException class (System)

11 ToString

12
13
14 *Description*

15 The exception that is thrown when a program contains an invalid IL or
16 metadata. Generally this indicates a bug in a compiler.

17 **System.InvalidProgramException** uses the HRESULT
18 COR_E_INVALIDPROGRAM, which has the value 0x8013153A.

19 InvalidProgramException

20 *Example Syntax:*

21 ToString

22
23 [C#] public InvalidProgramException();

24 [C++] public: InvalidProgramException();

25 [VB] Public Sub New()

[JScript] public function InvalidProgramException(); Initializes a new instance of the **System.InvalidProgramException** class.

Description

Initializes a new instance of the **System.InvalidProgramException** class with default properties.

The following table shows the initial property values for an instance of **System.InvalidProgramException**.

InvalidProgramException

Example Syntax:

ToString

[C#] public InvalidProgramException(string message);

[C++] public: InvalidProgramException(String* message);

[VB] Public Sub New(ByVal message As String)

[JScript] public function InvalidProgramException(message : String);

Description

Initializes a new instance of the **System.InvalidProgramException** class with a specified error message.

The following table shows the initial property values for an instance of **System.InvalidProgramException**. The error message that explains the reason for the exception.

InvalidProgramException

Example Syntax:

ToString

```
[C#] public InvalidProgramException(string message, Exception inner);  
[C++] public: InvalidProgramException(String* message, Exception* inner);  
[VB] Public Sub New(ByVal message As String, ByVal inner As Exception)  
[JScript] public function InvalidProgramException(message : String, inner :  
Exception);
```

Description

Initializes a new instance of the **System.InvalidProgramException** class with a specified error message and a reference to the inner exception that is the root cause of this exception.

When an **Exception** *X* is thrown as a direct result of a previous exception *Y*, the **System.Exception.InnerException** property of *X* should contain a reference to *Y*. The **InnerException** property returns the same value as was passed into the constructor, or **null** if the inner exception value was not supplied to the constructor. The error message that explains the reason for the exception. An instance of **System.Exception** that is the cause of the current **Exception**. If *inner* is non-null, then the current **Exception** is raised in a catch block handling *inner*.

HelpLink

HResult

InnerException

Message

Source

StackTrace

1 TargetSite

2 IServiceProvider interface (System)

3 ToString

4
5
6 *Description*

7 Defines a mechanism for retrieving a "service" object; that is, an object
8 which provides custom support to other objects.

9 This interface is implemented by a class or value type which provides a
10 service to other objects.

11 GetService

12
13 [C#] object GetService(Type serviceType);

14 [C++] Object* GetService(Type* serviceType);

15 [VB] Function GetService(ByVal serviceType As Type) As Object

16 [JScript] function GetService(serviceType : Type) : Object;

17
18 *Description*

19 Gets the service object of the specified type.

20 *Return Value:* A service object of type *serviceType* -or- **null** if there is no service
21 object of type *serviceType* . An object that specifies the type of service object to
22 get.

23 LoaderOptimization enumeration (System)

24 GetService

1
2
3 *Description*

4 An enumeration used with the **System.LoaderOptimizationAttribute**
5 class to specify loader optimizations for an executable.

6 GetService

7
8 [C#] public const LoaderOptimization MultiDomain;

9 [C++] public: const LoaderOptimization MultiDomain;

10 [VB] Public Const MultiDomain As LoaderOptimization

11 [JScript] public var MultiDomain : LoaderOptimization;

12
13 *Description*

14 Indicates that the application will probably have many domains which use
15 the same code, and the loader should share maximal internal resources across
16 application domains.

17 GetService

18
19 [C#] public const LoaderOptimization MultiDomainHost;

20 [C++] public: const LoaderOptimization MultiDomainHost;

21 [VB] Public Const MultiDomainHost As LoaderOptimization

22 [JScript] public var MultiDomainHost : LoaderOptimization;

23
24 *Description*
25

Indicates that the application will probably host unique code in multiple domains, and the loader should share resources across application domains for globally available (strong named) assemblies only.

GetService

[C#] public const LoaderOptimization NotSpecified;

[C++] public: const LoaderOptimization NotSpecified;

[VB] Public Const NotSpecified As LoaderOptimization

[JScript] public var NotSpecified : LoaderOptimization;

Description

Indicates no optimizations for sharing internal resources are specified. If the default domain or hosting interface specified an optimization then the loader uses that; otherwise, the loader uses **System.LoaderOptimization.SingleDomain**.

GetService

[C#] public const LoaderOptimization SingleDomain;

[C++] public: const LoaderOptimization SingleDomain;

[VB] Public Const SingleDomain As LoaderOptimization

[JScript] public var SingleDomain : LoaderOptimization;

Description

Indicates that the application will probably have a single domain, and loader should not share internal resources across application domains.

LoaderOptimizationAttribute class (System)

ToString

Description

Used to set the default loader optimization policy for the process. Should only be set on the main method for an application. It is ignored on all other methods.

The loader can make optimizations to share internal resource across application domains, at a slight expense in static access speed. This attribute tells the loader what type of application to optimize for - **SingleDomain** , **MultiDomain** (each domain running the same classes), or **MultiDomainHost** (multiple domains that can run different classes).

LoaderOptimizationAttribute

Example Syntax:

ToString

[C#] public LoaderOptimizationAttribute(byte value);

[C++] public: LoaderOptimizationAttribute(unsigned char value);

[VB] Public Sub New(ByVal value As Byte)

[JScript] public function LoaderOptimizationAttribute(value : Byte); Initializes a new instance of the **System.LoaderOptimizationAttribute** class.

Description

1 Initializes a new instance of the **System.LoaderOptimizationAttribute**
2 class to the specified value. A value equivalent to a **System.LoaderOptimization**
3 constant.

4 LoaderOptimizationAttribute

5 *Example Syntax:*

6 ToString

7
8 [C#] public LoaderOptimizationAttribute(LoaderOptimization value);

9 [C++] public: LoaderOptimizationAttribute(LoaderOptimization value);

10 [VB] Public Sub New(ByVal value As LoaderOptimization)

11 [JScript] public function LoaderOptimizationAttribute(value :

12 LoaderOptimization);

13
14 *Description*

15 Initializes a new instance of the **System.LoaderOptimizationAttribute**
16 class to the specified value. A **System.LoaderOptimization** constant.

17 TypeId

18 Value

19 ToString

20
21
22 *Description*

23 Gets the current **System.LoaderOptimization** value for this instance.

24 LocalDataStoreSlot class (System)

25 ToString

Description

Encapsulates a memory slot to store local data. This class cannot be inherited.

Threads and contexts use a local store memory mechanism to store thread-specific and context-specific data, respectively. The common language runtime allocates a multi-slot data store array to each process when it is created. The thread or context calls various functions to allocate a data slot in the data store, to store and retrieve a data value in the slot, and to free a data slot for reuse after the thread or context object expires.

Finalize

[C#] ~LocalDataStoreSlot();

[C++] ~LocalDataStoreSlot();

[VB] Overrides Protected Sub Finalize()

[JScript] protected override function Finalize();

Description

Releases the memory slot reserved by an object when the object no longer exists.

System.LocalDataStoreSlot.Finalize locks the data store manager before marking the data slot as unoccupied.

MarshalByRefObject class (System)

ToString

Description

Base class for Remoting objects that need to be marshal by reference. This includes WellKnown SingleCall and WellKnown Singleton WebService objects and Client Activated Objects.

Some key points to classes that derived from MarshalByRefObject: *

- Derive from System.MarshalByRefObject or any of its children (except from context bound objects which derive from System.ContextBoundObject).

MarshalByRefObject

Example Syntax:

ToString

[C#] protected MarshalByRefObject();

[C++] protected: MarshalByRefObject();

[VB] Protected Sub New()

[JScript] protected function MarshalByRefObject();

CreateObjRef

[C#] public virtual ObjRef CreateObjRef(Type requestedType);

[C++] public: virtual ObjRef* CreateObjRef(Type* requestedType);

[VB] Overridable Public Function CreateObjRef(ByVal requestedType As Type)

As ObjRef

[JScript] public function CreateObjRef(requestedType : Type) : ObjRef;

GetLifetimeService

1
2 [C#] public object GetLifetimeService();

3 [C++] public: __sealed Object* GetLifetimeService();

4 [VB] NotOverridable Public Function GetLifetimeService() As Object

5 [JScript] public function GetLifetimeService() : Object;

6
7 *Description*

8 Retrieves a lifetime service object that controls the lifetime policy for this
9 instance. For the default Lifetime service this will be an object of type ILease.

10 *Return Value:* Returns Object to control lifetime Service.

11 InitializeLifetimeService

12
13 [C#] public virtual object InitializeLifetimeService();

14 [C++] public: virtual Object* InitializeLifetimeService();

15 [VB] Overridable Public Function InitializeLifetimeService() As Object

16 [JScript] public function InitializeLifetimeService() : Object;

17
18 *Description*

19 Objects can provide their own lease and so control their own lifetime. They
20 do this by overriding the **InitializeLifetimeService** method provided on

21 **MarshalByRefObject** .

22 Math class (System)

23 ToString

1
2
3 *Description*

4 Provides constants and static methods for trigonometric, logarithmic, and
5 other common mathematical functions.

6 ToString

7
8 [C#] public const double E;

9 [C++] public: const double E;

10 [VB] Public Const E As Double

11 [JScript] public var E : double;

12
13 *Description*

14 A constant, **e** , that specifies the natural logarithmic base.

15 The value of this field is 2.7182818284590452354.

16 ToString

17
18 [C#] public const double PI;

19 [C++] public: const double PI;

20 [VB] Public Const PI As Double

21 [JScript] public var PI : double;

22
23 *Description*

24 A constant, (**pi**), that specifies the ratio of the circumference of a circle to
25 its diameter.

The value of this field is 3.14159265358979323846.

Abs

[C#] public static decimal Abs(decimal value);

[C++] public: static Decimal Abs(Decimal value);

[VB] Public Shared Function Abs(ByVal value As Decimal) As Decimal

[JScript] public static function Abs(value : Decimal) : Decimal;

Description

Returns the absolute value of a **Decimal** number.

Return Value: A **Decimal** , x, such that 0 (<=) x (<=) **System.Decimal.MaxValue**

. A number in the range **System.Decimal.MinValue** (<=) value (<=)

System.Decimal.MaxValue.

Abs

[C#] public static double Abs(double value);

[C++] public: static double Abs(double value);

[VB] Public Shared Function Abs(ByVal value As Double) As Double

[JScript] public static function Abs(value : double) : double;

Description

Returns the absolute value of a double-precision floating point number.

Return Value: A double-precision floating point number, x, such that 0 (<=) x (<=)

System.Double.MaxValue . A number in the range **System.Double.MinValue** <

value (<=) **System.Double.MaxValue**.

Abs

```
[C#] public static short Abs(short value);  
[C++] public: static short Abs(short value);  
[VB] Public Shared Function Abs(ByVal value As Short) As Short  
[JScript] public static function Abs(value : Int16) : Int16;
```

Description

Returns the absolute value of a 16-bit signed integer.

Return Value: A 16-bit signed integer, x, such that 0 (<=) x (<=)

System.Int16.MaxValue . A number in the range **System.Int16.MinValue** < value (<=) **System.Int16.MaxValue**.

Abs

```
[C#] public static int Abs(int value);  
[C++] public: static int Abs(int value);  
[VB] Public Shared Function Abs(ByVal value As Integer) As Integer  
[JScript] public static function Abs(value : int) : int;
```

Description

Returns the absolute value of a 32-bit signed integer.

Return Value: A 32-bit signed integer, x, such that 0 (<=) x (<=)

System.Int32.MaxValue . A number in the range **System.Int32.MinValue** < value (<=) **System.Int32.MaxValue**.

Abs

1
2 [C#] public static long Abs(long value);

3 [C++] public: static __int64 Abs(__int64 value);

4 [VB] Public Shared Function Abs(ByVal value As Long) As Long

5 [JScript] public static function Abs(value : long) : long;

6
7 *Description*

8 Returns the absolute value of a 64-bit signed integer.

9 *Return Value:* A 64-bit signed integer, x, such that 0 (\leq) x (\leq)

10 **System.Int64.MaxValue** . A number in the range **System.Int64.MinValue** <
11 *value* (\leq) **System.Int64.MaxValue**.

12 Abs

13
14 [C#] public static sbyte Abs(sbyte value);

15 [C++] public: static char Abs(char value);

16 [VB] Public Shared Function Abs(ByVal value As SByte) As SByte

17 [JScript] public static function Abs(value : SByte) : SByte; Returns the absolute
18 value of a specified number.

19
20 *Description*

21 Returns the absolute value of an 8-bit signed integer.

22 *Return Value:* An 8-bit signed integer, x, such that 0 (\leq) x (\leq)

23 **System.SByte.MaxValue** . A number in the range **System.SByte.MinValue** <
24 *value* (\leq) **System.SByte.MaxValue**.

25 Abs

1
2 [C#] public static float Abs(float value);

3 [C++] public: static float Abs(float value);

4 [VB] Public Shared Function Abs(ByVal value As Single) As Single

5 [JScript] public static function Abs(value : float) : float;

6
7 *Description*

8 Returns the absolute value of a single-precision floating point number.

9 *Return Value:* A single-precision floating point number, x, such that $0 \leq x \leq$

10 **System.Single.MaxValue** . A number in the range **System.Single.MinValue** <
11 *value* \leq **System.Single.MaxValue**.

12 **Acos**

13
14 [C#] public static double Acos(double d);

15 [C++] public: static double Acos(double d);

16 [VB] Public Shared Function Acos(ByVal d As Double) As Double

17 [JScript] public static function Acos(d : double) : double;

18
19 *Description*

20 Returns the angle whose cosine is the specified number.

21 *Return Value:* An angle, q, measured in radians, such that $0 \leq q \leq \pi$ -or-

22 **System.Double.NaN** if $d < -1$ or $d > 1$.

23 Multiply the return value by $180/\pi$ to convert from radians to degrees. A
24 number representing a cosine, where $-1 \leq d \leq 1$.

25 **Asin**

```

1
2 [C#] public static double Asin(double d);
3 [C++] public: static double Asin(double d);
4 [VB] Public Shared Function Asin(ByVal d As Double) As Double
5 [JScript] public static function Asin(d : double) : double;
6

```

7 *Description*

8 Returns the angle whose sine is the specified number.

9 *Return Value:* An angle, q, measured in radians, such that $-(\pi)/2 \leq q \leq (\pi)/2$
10 -or- **System.Double.NaN** if $d < -1$ or $d > 1$.

11 A positive return value represents a counterclockwise angle from the x-
12 axis; a negative return value represents a clockwise angle. A number representing
13 a sine, where $-1 \leq d \leq 1$.

14 *Atan*

```

15
16 [C#] public static double Atan(double d);
17 [C++] public: static double Atan(double d);
18 [VB] Public Shared Function Atan(ByVal d As Double) As Double
19 [JScript] public static function Atan(d : double) : double;
20

```

21 *Description*

22 Returns the angle whose tangent is the specified number.

23 *Return Value:* An angle, q, measured in radians, such that $-(\pi)/2 \leq q \leq$
24 $(\pi)/2$.

1 A positive return value represents a counterclockwise angle from the x-
2 axis; a negative return value represents a clockwise angle. A number representing
3 a tangent.

4 Atan2

5
6 [C#] public static double Atan2(double y, double x);

7 [C++] public: static double Atan2(double y, double x);

8 [VB] Public Shared Function Atan2(ByVal y As Double, ByVal x As Double) As
9 Double

10 [JScript] public static function Atan2(y : double, x : double) : double;

11 12 *Description*

13 Returns the angle whose tangent is the quotient of two specified numbers.

14 *Return Value:* An angle, q , measured in radians, such that $-(\pi) < q \leq (\pi)$, and
15 $\tan(q) = y / x$, where (x, y) is a point in the Cartesian plane. Observe the
16 following: For (x, y) in quadrant 1, $0 < q < (\pi)/2$.

17 The return value is the angle in the Cartesian plane formed by the x-axis,
18 and a vector starting from the origin, (0,0), and terminating at the point, (x, y) .

19 The y coordinate of a point. The x coordinate of a point.

20 Ceiling

21
22 [C#] public static double Ceiling(double a);

23 [C++] public: static double Ceiling(double a);

24 [VB] Public Shared Function Ceiling(ByVal a As Double) As Double

25 [JScript] public static function Ceiling(a : double) : double;

1
2 *Description*

3 Returns the smallest whole number greater than or equal to the specified
4 number.

5 *Return Value:* The smallest whole number greater than or equal to a .

6 The behavior of this method follows IEEE Standard 754, section 4. This
7 kind of rounding is sometimes called rounding towards positive infinity. A
8 number.

9 Cos

10
11 [C#] public static double Cos(double d);

12 [C++] public: static double Cos(double d);

13 [VB] Public Shared Function Cos(ByVal d As Double) As Double

14 [JScript] public static function Cos(d : double) : double;

15
16 *Description*

17 Returns the cosine of the specified angle.

18 *Return Value:* The cosine of d .

19 The angle, d , must be in radians. Multiply by (pi)/180 to convert degrees to
20 radians. An angle, measured in radians.

21 Cosh

22
23 [C#] public static double Cosh(double value);

24 [C++] public: static double Cosh(double value);

25 [VB] Public Shared Function Cosh(ByVal value As Double) As Double

1 [JScript] public static function Cosh(value : double) : double;

3 *Description*

4 Returns the hyperbolic cosine of the specified angle.

5 *Return Value:* The hyperbolic cosine of *value* . An angle, measured in radians.

6 **Exp**

8 [C#] public static double Exp(double d);

9 [C++] public: static double Exp(double d);

10 [VB] Public Shared Function Exp(ByVal d As Double) As Double

11 [JScript] public static function Exp(d : double) : double;

13 *Description*

14 Returns **e** raised to the specified power.

15 *Return Value:* The number **e** raised to the power *d* .

16 Use the **System.Math.Pow(System.Double, System.Double)** method to
17 calculate powers of other bases. A number specifying a power.

18 **Floor**

20 [C#] public static double Floor(double d);

21 [C++] public: static double Floor(double d);

22 [VB] Public Shared Function Floor(ByVal d As Double) As Double

23 [JScript] public static function Floor(d : double) : double;

25 *Description*

1 Returns the largest whole number less than or equal to the specified
2 number.

3 *Return Value:* The largest whole number less than or equal to d .

4 The behavior of this method follows IEEE Standard 754, section 4. This
5 kind of rounding is sometimes called rounding towards negative infinity. A
6 number.

7 IEEERemainder

8
9 [C#] public static double IEEERemainder(double x, double y);

10 [C++] public: static double IEEERemainder(double x, double y);

11 [VB] Public Shared Function IEEERemainder(ByVal x As Double, ByVal y As
12 Double) As Double

13 [JScript] public static function IEEERemainder(x : double, y : double) : double;

14 15 Description

16 Returns the remainder resulting from the division of a specified number by
17 another specified number.

18 *Return Value:* A number equal to $x - (y Q)$, where Q is the quotient of x / y
19 rounded to the nearest integer (if x / y falls halfway between two integers, the even
20 integer is returned).

21 This operation complies with the remainder operation defined in Section
22 5.1 of ANSI/IEEE Std 754-1985; IEEE Standard for Binary Floating-Point
23 Arithmetic; Institute of Electrical and Electronics Engineers, Inc; 1985. A
24 dividend. A divisor.

25 Log

[C#] public static double Log(double d);

[C++] public: static double Log(double d);

[VB] Public Shared Function Log(ByVal d As Double) As Double

[JScript] public static function Log(d : double) : double; Returns the logarithm of a specified number.

Description

Returns the natural (base **e**) logarithm of a specified number.

Return Value: Sign of *d* Returns Positive The natural logarithm of *d* ; that is, $\ln d$, or $\log_e d$ Zero **System.Double.PositiveInfinity** Negative **System.Double.NaN**

d is specified as a base 10 number. A number whose logarithm is to be found.

Log

[C#] public static double Log(double a, double newBase);

[C++] public: static double Log(double a, double newBase);

[VB] Public Shared Function Log(ByVal a As Double, ByVal newBase As Double) As Double

[JScript] public static function Log(a : double, newBase : double) : double;

Description

Returns the logarithm of a specified number in a specified base.

Return Value: Sign of *d* Returns Positive The logarithm of *a* , in base, *newBase* ; that is, \log_a .

1 *a* and *newBase* are specified as base 10 numbers. A number whose
2 logarithm is to be found. The base of the logarithm.

3 Log10

4
5 [C#] public static double Log10(double d);

6 [C++] public: static double Log10(double d);

7 [VB] Public Shared Function Log10(ByVal d As Double) As Double

8 [JScript] public static function Log10(d : double) : double;

9 10 *Description*

11 Returns the base 10 logarithm of a specified number.

12 *Return Value:* Sign of *d* Returns Positive The base 10 log of *d* ; that is, $\log d$.

13 *d* is specified as a base 10 number. A number whose logarithm is to be
14 found.

15 Max

16
17 [C#] public static byte Max(byte val1, byte val2);

18 [C++] public: static unsigned char Max(unsigned char val1, unsigned char val2);

19 [VB] Public Shared Function Max(ByVal val1 As Byte, ByVal val2 As Byte) As

20 Byte

21 [JScript] public static function Max(val1 : Byte, val2 : Byte) : Byte;

22 23 *Description*

1 Returns the larger of two 8-bit unsigned integers.

2 *Return Value:* *val1* or *val2* , whichever is larger. The first of two 8-bit unsigned
3 integers to compare. The second of two 8-bit unsigned integers to compare.

4 Max

5
6 [C#] public static decimal Max(decimal val1, decimal val2);

7 [C++] public: static Decimal Max(Decimal val1, Decimal val2);

8 [VB] Public Shared Function Max(ByVal val1 As Decimal, ByVal val2 As
9 Decimal) As Decimal

10 [JScript] public static function Max(val1 : Decimal, val2 : Decimal) : Decimal;

11
12 *Description*

13 Returns the larger of two **Decimal** numbers.

14 *Return Value:* *val1* or *val2* , whichever is larger. The first of two **System.Decimal**
15 numbers to compare. The second of two **System.Decimal** numbers to compare.

16 Max

17
18 [C#] public static double Max(double val1, double val2);

19 [C++] public: static double Max(double val1, double val2);

20 [VB] Public Shared Function Max(ByVal val1 As Double, ByVal val2 As Double)
21 As Double

22 [JScript] public static function Max(val1 : double, val2 : double) : double;

23
24 *Description*

1 Returns the larger of two double-precision floating point numbers.

2 *Return Value:* *val1* or *val2* , whichever is larger. The first of two double-precision
3 floating point numbers to compare. The second of two double-precision floating
4 point numbers to compare.

5 Max

6
7 [C#] public static short Max(short val1, short val2);

8 [C++] public: static short Max(short val1, short val2);

9 [VB] Public Shared Function Max(ByVal val1 As Short, ByVal val2 As Short) As

10 Short

11 [JScript] public static function Max(val1 : Int16, val2 : Int16) : Int16;

12
13 *Description*

14 Returns the larger of two 16-bit signed integers.

15 *Return Value:* *val1* or *val2* , whichever is larger. The first of two 16-bit signed
16 integers to compare. The second of two 16-bit signed integers to compare.

17 Max

18
19 [C#] public static int Max(int val1, int val2);

20 [C++] public: static int Max(int val1, int val2);

21 [VB] Public Shared Function Max(ByVal val1 As Integer, ByVal val2 As Integer)

22 As Integer

23 [JScript] public static function Max(val1 : int, val2 : int) : int;

24
25 *Description*

1 Returns the larger of two 32-bit signed integers.

2 *Return Value:* *val1* or *val2* , whichever is larger. The first of two 32-bit signed
3 integers to compare. The second of two 32-bit signed integers to compare.

4 Max

5
6 [C#] public static long Max(long val1, long val2);

7 [C++] public: static __int64 Max(__int64 val1, __int64 val2);

8 [VB] Public Shared Function Max(ByVal val1 As Long, ByVal val2 As Long) As

9 Long

10 [JScript] public static function Max(val1 : long, val2 : long) : long;

11
12 *Description*

13 Returns the larger of two 64-bit signed integers.

14 *Return Value:* *val1* or *val2* , whichever is larger. The first of two 64-bit signed
15 integers to compare. The second of two 64-bit signed integers to compare.

16 Max

17
18 [C#] public static sbyte Max(sbyte val1, sbyte val2);

19 [C++] public: static char Max(char val1, char val2);

20 [VB] Public Shared Function Max(ByVal val1 As SByte, ByVal val2 As SByte)

21 As SByte

22 [JScript] public static function Max(val1 : SByte, val2 : SByte) : SByte; Returns
23 the larger of two specified numbers.

24
25 *Description*

1 Returns the larger of two 8-bit signed integers.

2 *Return Value:* *val1* or *val2* , whichever is larger. The first of two 8-bit unsigned
3 integers to compare. The second of two 8-bit unsigned integers to compare.

4 Max

5
6 [C#] public static float Max(float val1, float val2);

7 [C++] public: static float Max(float val1, float val2);

8 [VB] Public Shared Function Max(ByVal val1 As Single, ByVal val2 As Single)

9 As Single

10 [JScript] public static function Max(val1 : float, val2 : float) : float;

11
12 *Description*

13 Returns the larger of two single-precision floating point numbers.

14 *Return Value:* *val1* or *val2* , whichever is larger. The first of two single-precision
15 floating point numbers to compare. The second of two single-precision floating
16 point numbers to compare.

17 Max

18
19 [C#] public static ushort Max(ushort val1, ushort val2);

20 [C++] public: static unsigned short Max(unsigned short val1, unsigned short val2);

21 [VB] Public Shared Function Max(ByVal val1 As UInt16, ByVal val2 As UInt16)

22 As UInt16

23 [JScript] public static function Max(val1 : UInt16, val2 : UInt16) : UInt16;

24
25 *Description*

1 Returns the larger of two 16-bit unsigned integers.

2 *Return Value:* *val1* or *val2* , whichever is larger. The first of two 16-bit unsigned
3 integers to compare. The second of two 16-bit unsigned integers to compare.

4 Max

5
6 [C#] public static uint Max(uint val1, uint val2);

7 [C++] public: static unsigned int Max(unsigned int val1, unsigned int val2);

8 [VB] Public Shared Function Max(ByVal val1 As UInt32, ByVal val2 As UInt32)
9 As UInt32

10 [JScript] public static function Max(val1 : UInt32, val2 : UInt32) : UInt32;

11
12 *Description*

13 Returns the larger of two 32-bit unsigned integers.

14 *Return Value:* *val1* or *val2* , whichever is larger. The first of two 32-bit unsigned
15 integers to compare. The second of two 32-bit unsigned integers to compare.

16 Max

17
18 [C#] public static ulong Max(ulong val1, ulong val2);

19 [C++] public: static unsigned __int64 Max(unsigned __int64 val1, unsigned
20 __int64 val2);

21 [VB] Public Shared Function Max(ByVal val1 As UInt64, ByVal val2 As UInt64)
22 As UInt64

23 [JScript] public static function Max(val1 : UInt64, val2 : UInt64) : UInt64;

24
25 *Description*

1 Returns the larger of two 64-bit unsigned integers.

2 *Return Value:* *val1* or *val2* , whichever is larger. The first of two 64-bit unsigned
3 integers to compare. The second of two 64-bit unsigned integers to compare.

4 Min

5
6 [C#] public static byte Min(byte val1, byte val2);

7 [C++] public: static unsigned char Min(unsigned char val1, unsigned char val2);

8 [VB] Public Shared Function Min(ByVal val1 As Byte, ByVal val2 As Byte) As

9 Byte

10 [JScript] public static function Min(val1 : Byte, val2 : Byte) : Byte;

11
12 *Description*

13 Returns the smaller of two 8-bit unsigned integers.

14 *Return Value:* *val1* or *val2* , whichever is smaller. The first of two 8-bit unsigned
15 integers to compare. The second of two 8-bit unsigned integers to compare.

16 Min

17
18 [C#] public static decimal Min(decimal val1, decimal val2);

19 [C++] public: static Decimal Min(Decimal val1, Decimal val2);

20 [VB] Public Shared Function Min(ByVal val1 As Decimal, ByVal val2 As
21 Decimal) As Decimal

22 [JScript] public static function Min(val1 : Decimal, val2 : Decimal) : Decimal;

23
24 *Description*

1 Returns the smaller of two **Decimal** numbers.

2 *Return Value:* *a* or *val2* , whichever is smaller. The first of two **System.Decimal**
3 numbers to compare. The second of two **System.Decimal** numbers to compare.

4 Min

5
6 [C#] public static double Min(double val1, double val2);

7 [C++] public: static double Min(double val1, double val2);

8 [VB] Public Shared Function Min(ByVal val1 As Double, ByVal val2 As Double)
9 As Double

10 [JScript] public static function Min(val1 : double, val2 : double) : double;

11
12 *Description*

13 Returns the smaller of two double-precision floating point numbers.

14 *Return Value:* *a* or *val2* , whichever is smaller. The first of two double-precision
15 floating point numbers to compare. The second of two double-precision floating
16 point numbers to compare.

17 Min

18
19 [C#] public static short Min(short val1, short val2);

20 [C++] public: static short Min(short val1, short val2);

21 [VB] Public Shared Function Min(ByVal val1 As Short, ByVal val2 As Short) As
22 Short

23 [JScript] public static function Min(val1 : Int16, val2 : Int16) : Int16;

24
25 *Description*

1 Returns the smaller of two 16-bit signed integers.

2 *Return Value:* *val1* or *val2* , whichever is smaller. The first of two 16-bit signed
3 integers to compare. The second of two 16-bit signed integers to compare.

4 Min

5
6 [C#] public static int Min(int val1, int val2);

7 [C++] public: static int Min(int val1, int val2);

8 [VB] Public Shared Function Min(ByVal val1 As Integer, ByVal val2 As Integer)

9 As Integer

10 [JScript] public static function Min(val1 : int, val2 : int) : int;

11
12 *Description*

13 Returns the smaller of two 32-bit signed integers.

14 *Return Value:* *val1* or *val2* , whichever is smaller. The first of two 32-bit signed
15 integers to compare. The second of two 32-bit signed integers to compare.

16 Min

17
18 [C#] public static long Min(long val1, long val2);

19 [C++] public: static __int64 Min(__int64 val1, __int64 val2);

20 [VB] Public Shared Function Min(ByVal val1 As Long, ByVal val2 As Long) As

21 Long

22 [JScript] public static function Min(val1 : long, val2 : long) : long;

23
24 *Description*

1 Returns the smaller of two 64-bit signed integers.

2 *Return Value:* *val1* or *val2* , whichever is smaller. The first of two 64-bit signed
3 integers to compare. The second of two 64-bit signed integers to compare.

4 Min

5
6 [C#] public static sbyte Min(sbyte val1, sbyte val2);

7 [C++] public: static char Min(char val1, char val2);

8 [VB] Public Shared Function Min(ByVal val1 As SByte, ByVal val2 As SByte)

9 As SByte

10 [JScript] public static function Min(val1 : SByte, val2 : SByte) : SByte; Returns
11 the smaller of two numbers.

12
13 *Description*

14 Returns the smaller of two 8-bit signed integers.

15 *Return Value:* *val1* or *val2* , whichever is smaller. The first of two 8-bit signed
16 integers to compare. The second of two 8-bit signed integers to compare.

17 Min

18
19 [C#] public static float Min(float val1, float val2);

20 [C++] public: static float Min(float val1, float val2);

21 [VB] Public Shared Function Min(ByVal val1 As Single, ByVal val2 As Single)

22 As Single

23 [JScript] public static function Min(val1 : float, val2 : float) : float;

24
25 *Description*

1 Returns the smaller of two single-precision floating point numbers.

2 *Return Value:* *val1* or *val2* , whichever is smaller. The first of two single-precision
3 floating point numbers to compare. The second of two single-precision floating
4 point numbers to compare.

5 Min

6
7 [C#] public static ushort Min(ushort val1, ushort val2);

8 [C++] public: static unsigned short Min(unsigned short val1, unsigned short val2);

9 [VB] Public Shared Function Min(ByVal val1 As UInt16, ByVal val2 As UInt16)

10 As UInt16

11 [JScript] public static function Min(val1 : UInt16, val2 : UInt16) : UInt16;

12
13 *Description*

14 Returns the smaller of two 16-bit unsigned integers.

15 *Return Value:* *val1* or *val2* , whichever is smaller. The first of two 16-bit unsigned
16 integers to compare. The second of two 16-bit unsigned integers to compare.

17 Min

18
19 [C#] public static uint Min(uint val1, uint val2);

20 [C++] public: static unsigned int Min(unsigned int val1, unsigned int val2);

21 [VB] Public Shared Function Min(ByVal val1 As UInt32, ByVal val2 As UInt32)

22 As UInt32

23 [JScript] public static function Min(val1 : UInt32, val2 : UInt32) : UInt32;

24
25 *Description*

1 Returns the smaller of two 32-bit unsigned integers.

2 *Return Value:* *val1* or *val2* , whichever is smaller. The first of two 32-bit unsigned
3 integers to compare. The second of two 32-bit unsigned integers to compare.

4 Min

5
6 [C#] public static ulong Min(ulong val1, ulong val2);

7 [C++] public: static unsigned __int64 Min(unsigned __int64 val1, unsigned
8 __int64 val2);

9 [VB] Public Shared Function Min(ByVal val1 As UInt64, ByVal val2 As UInt64)
10 As UInt64

11 [JScript] public static function Min(val1 : UInt64, val2 : UInt64) : UInt64;

12
13 *Description*

14 Returns the smaller of two 64-bit unsigned integers.

15 *Return Value:* *val1* or *val2* , whichever is smaller. The first of two 64-bit unsigned
16 integers to compare. The second of two 64-bit unsigned integers to compare.

17 Pow

18
19 [C#] public static double Pow(double x, double y);

20 [C++] public: static double Pow(double x, double y);

21 [VB] Public Shared Function Pow(ByVal x As Double, ByVal y As Double) As
22 Double

23 [JScript] public static function Pow(x : double, y : double) : double;

24
25 *Description*

1 Returns a specified number raised to the specified power.

2 *Return Value:* The number x raised to the power y . A number to be raised to a
3 power. A number that specifies a power.

4 Round

5
6 [C#] public static decimal Round(decimal d);

7 [C++] public: static Decimal Round(Decimal d);

8 [VB] Public Shared Function Round(ByVal d As Decimal) As Decimal

9 [JScript] public static function Round(d : Decimal) : Decimal;

10
11 *Description*

12 Returns the whole number nearest the specified value.

13 *Return Value:* The whole number nearest parameter d . If d is halfway between
14 two whole numbers, one of which by definition is even and the other odd, then the
15 even number is returned.

16 The behavior of this method follows IEEE Standard 754, section 4. This
17 kind of rounding is sometimes called rounding to nearest, or banker's rounding. A
18 **System.Decimal** number to be rounded.

19 Round

20
21 [C#] public static double Round(double a);

22 [C++] public: static double Round(double a);

23 [VB] Public Shared Function Round(ByVal a As Double) As Double

24 [JScript] public static function Round(a : double) : double; Returns the number
25 nearest the specified value.

Description

Returns the whole number nearest the specified value.

Return Value: The whole number nearest a . If a is halfway between two whole numbers, one of which by definition is even and the other odd, then the even number is returned.

The behavior of this method follows IEEE Standard 754, section 4. This kind of rounding is sometimes called rounding to nearest, or banker's rounding. A double-precision floating point number to be rounded.

Round

[C#] public static decimal Round(decimal d, int decimals);

[C++] public: static Decimal Round(Decimal d, int decimals);

[VB] Public Shared Function Round(ByVal d As Decimal, ByVal decimals As Integer) As Decimal

[JScript] public static function Round(d : Decimal, decimals : int) : Decimal;

Description

Returns the number with the specified precision nearest the specified value.

Return Value: The number nearest d with precision equal to $decimals$. If d is halfway between two numbers, one of which is even and the other odd, then the even number is returned. If the precision of d is less than $decimals$, then d is returned unchanged.

The *decimals* parameter specifies the number of significant fractional digits in the return value and ranges from 0 to 28. If *decimals* is zero, then a whole

number is returned. A **System.Decimal** number to be rounded. The number of significant fractional digits (precision) in the return value.

Round

[C#] public static double Round(double value, int digits);

[C++] public: static double Round(double value, int digits);

[VB] Public Shared Function Round(ByVal value As Double, ByVal digits As Integer) As Double

[JScript] public static function Round(value : double, digits : int) : double;

Description

Returns the number with the specified precision nearest the specified value.

Return Value: The number nearest *value* with precision equal to *digits* . If *value* is halfway between two numbers, one of which is even and the other odd, then the even number is returned. If the precision of *value* is less than *digits* , then *value* is returned unchanged.

The *digits* parameter specifies the number of significant fractional digits in the return value and ranges from 0 to 15. If *digits* is zero, then a whole number is returned. A double-precision floating point number to be rounded. The number of significant fractional digits (precision) in the return value.

Sign

[C#] public static int Sign(decimal value);

[C++] public: static int Sign(Decimal value);

[VB] Public Shared Function Sign(ByVal value As Decimal) As Integer

1 [JScript] public static function Sign(value : Decimal) : int;

3 *Description*

4 Returns a value indicating the sign of a **Decimal** number.

5 *Return Value:* A number indicating the sign of *value* . A signed **System.Decimal**
6 number.

7 Sign

9 [C#] public static int Sign(double value);

10 [C++] public: static int Sign(double value);

11 [VB] Public Shared Function Sign(ByVal value As Double) As Integer

12 [JScript] public static function Sign(value : double) : int;

14 *Description*

15 Returns a value indicating the sign of a double-precision floating point
16 number.

17 *Return Value:* A number indicating the sign of *value* . A signed number.

18 Sign

20 [C#] public static int Sign(short value);

21 [C++] public: static int Sign(short value);

22 [VB] Public Shared Function Sign(ByVal value As Short) As Integer

23 [JScript] public static function Sign(value : Int16) : int; Returns a value indicating
24 the sign of a number.

1
2 *Description*

3 Returns a value indicating the sign of a 16-bit signed integer.

4 *Return Value:* A number indicating the sign of *value* . A signed number.

5 Sign

6
7 [C#] public static int Sign(int value);

8 [C++] public: static int Sign(int value);

9 [VB] Public Shared Function Sign(ByVal value As Integer) As Integer

10 [JScript] public static function Sign(value : int) : int; Returns a value indicating the
11 sign of a number.

12
13 *Description*

14 Returns a value indicating the sign of a 32-bit signed integer.

15 *Return Value:* A number indicating the sign of *value* . A signed number.

16 Sign

17
18 [C#] public static int Sign(long value);

19 [C++] public: static int Sign(__int64 value);

20 [VB] Public Shared Function Sign(ByVal value As Long) As Integer

21 [JScript] public static function Sign(value : long) : int;

22
23 *Description*

24 Returns a value indicating the sign of a 64-bit signed integer.

25 *Return Value:* A number indicating the sign of *value* . A signed number.

Sign

[C#] public static int Sign(sbyte value);

[C++] public: static int Sign(char value);

[VB] Public Shared Function Sign(ByVal value As SByte) As Integer

[JScript] public static function Sign(value : SByte) : int; Returns a value indicating the sign of a number.

Description

Returns a value indicating the sign of an 8-bit signed integer.

Return Value: A number indicating the sign of *value*. Number Description -1 *value* is less than zero. A signed number.

Sign

[C#] public static int Sign(float value);

[C++] public: static int Sign(float value);

[VB] Public Shared Function Sign(ByVal value As Single) As Integer

[JScript] public static function Sign(value : float) : int;

Description

Returns a value indicating the sign of a single-precision floating point number.

Return Value: A number indicating the sign of *value* . A signed number.

Sin

1
2 [C#] public static double Sin(double a);
3 [C++] public: static double Sin(double a);
4 [VB] Public Shared Function Sin(ByVal a As Double) As Double
5 [JScript] public static function Sin(a : double) : double;
6

7 *Description*

8 Returns the sine of the specified angle.

9 *Return Value:* The sine of *a* .

10 The angle, *a* , must be in radians. Multiply by (pi)/180 to convert degrees to
11 radians. An angle, measured in radians.

12 **Sinh**

13
14 [C#] public static double Sinh(double value);
15 [C++] public: static double Sinh(double value);
16 [VB] Public Shared Function Sinh(ByVal value As Double) As Double
17 [JScript] public static function Sinh(value : double) : double;
18

19 *Description*

20 Returns the hyperbolic sine of the specified angle.

21 *Return Value:* The hyperbolic sine of *value* . An angle, measured in radians.

22 **Sqrt**

23
24 [C#] public static double Sqrt(double d);
25 [C++] public: static double Sqrt(double d);

1 [VB] Public Shared Function Sqrt(ByVal d As Double) As Double

2 [JScript] public static function Sqrt(d : double) : double;

3
4 *Description*

5 Returns the square root of a specified number.

6 *Return Value:* Value of d Returns Zero, or positive The positive square root of d .

7 A number.

8 Tan

9
10 [C#] public static double Tan(double a);

11 [C++] public: static double Tan(double a);

12 [VB] Public Shared Function Tan(ByVal a As Double) As Double

13 [JScript] public static function Tan(a : double) : double;

14
15 *Description*

16 Returns the tangent of the specified angle.

17 *Return Value:* The tangent of a .

18 The angle, a , must be in radians. Multiply by (pi)/180 to convert degrees to
19 radians. An angle, measured in radians.

20 Tanh

21
22 [C#] public static double Tanh(double value);

23 [C++] public: static double Tanh(double value);

24 [VB] Public Shared Function Tanh(ByVal value As Double) As Double

25 [JScript] public static function Tanh(value : double) : double;

1
2 *Description*

3 Returns the hyperbolic tangent of the specified angle.

4 *Return Value:* The hyperbolic tangent of *value* . An angle, measured in radians.

5 MemberAccessException class (System)

6 ToString

7
8
9 *Description*

10 The exception that is thrown when an attempt to access a class member
11 fails.

12 **System.MemberAccessException** is the base class for
13 **System.FieldAccessException** , **System.MethodAccessException** ,
14 **System.MissingMemberException** , **System.MissingMethodException** , and
15 **System.MissingFieldException** . These exceptions are thrown when a class
16 member is not found or access to the member is not permitted.

17 MemberAccessException

18 *Example Syntax:*

19 ToString

20
21 [C#] public MemberAccessException();

22 [C++] public: MemberAccessException();

23 [VB] Public Sub New()

24 [JScript] public function MemberAccessException(); Initializes a new instance of
25 the **System.MemberAccessException** class.

Description

Initializes a new instance of the **System.MemberAccessException** class with default properties.

The following table shows the initial property values for an instance of **System.MemberAccessException**.

MemberAccessException

Example Syntax:

ToString

[C#] public MemberAccessException(string message);

[C++] public: MemberAccessException(String* message);

[VB] Public Sub New(ByVal message As String)

[JScript] public function MemberAccessException(message : String);

Description

Initializes a new instance of the **System.MemberAccessException** class with a specified error message.

The following table shows the initial property values for an instance of **System.MemberAccessException**. The error message that explains the reason for the exception.

MemberAccessException

Example Syntax:

ToString

1
2 [C#] protected MemberAccessException(SerializationInfo info, StreamingContext
3 context);

4 [C++] protected: MemberAccessException(SerializationInfo* info,
5 StreamingContext context);

6 [VB] Protected Sub New(ByVal info As SerializationInfo, ByVal context As
7 StreamingContext)

8 [JScript] protected function MemberAccessException(info : SerializationInfo,
9 context : StreamingContext);

10
11 *Description*

12 Initializes a new instance of the **System.MemberAccessException** class
13 with serialized data.

14 This constructor is called during deserialization to reconstitute the
15 exception object transmitted over a stream. For more information, see . The object
16 that holds the serialized object data. The contextual information about the source
17 or destination.

18 MemberAccessException

19 *Example Syntax:*

20 ToString

21
22 [C#] public MemberAccessException(string message, Exception inner);

23 [C++] public: MemberAccessException(String* message, Exception* inner);

24 [VB] Public Sub New(ByVal message As String, ByVal inner As Exception)

25 [JScript] public function MemberAccessException(message : String, inner :

Exception);

Description

Initializes a new instance of the **System.MemberAccessException** class with a specified error message and a reference to the inner exception that is the root cause of this exception.

When an **Exception** *X* is thrown as a direct result of a previous exception *Y*, the **System.Exception.InnerException** property of *X* should contain a reference to *Y*. The **InnerException** property returns the same value as was passed into the constructor, or **null** if the inner exception value was not supplied to the constructor. The error message that explains the reason for the exception. An instance of **System.Exception** that is the cause of the current **Exception**. If *inner* is non-null, then the current **Exception** is raised in a catch block handling *inner*.

HelpLink

HResult

InnerException

Message

Source

StackTrace

TargetSite

MethodAccessException class (System)

ToString

Description

The exception that is thrown when there is an illegal attempt to access a private or protected method inside a class.

System.MethodAccessException uses the HRESULT **COR_E_METHODACCESS**, which has the value 0x80131510.

MethodAccessException

Example Syntax:

ToString

[C#] public MethodAccessException();

[C++] public: MethodAccessException();

[VB] Public Sub New()

[JScript] public function MethodAccessException(); Initializes a new instance of the **System.MethodAccessException** class.

Description

Initializes a new instance of the **System.MethodAccessException** class with default properties.

The following table shows the initial property values for an instance of **System.MethodAccessException**.

MethodAccessException

Example Syntax:

ToString

[C#] public MethodAccessException(string message);

[C++] public: MethodAccessException(String* message);

1 [VB] Public Sub New(ByVal message As String)

2 [JScript] public function MethodAccessException(message : String);

3
4 *Description*

5 Initializes a new instance of the **System.MethodAccessException** class
6 with a specified error message.

7 The following table shows the initial property values for an instance of
8 **System.MethodAccessException** . The error message that explains the reason for
9 the exception.

10 MethodAccessException

11 *Example Syntax:*

12 ToString

13
14 [C#] protected MethodAccessException(SerializationInfo info, StreamingContext
15 context);

16 [C++] protected: MethodAccessException(SerializationInfo* info,
17 StreamingContext context);

18 [VB] Protected Sub New(ByVal info As SerializationInfo, ByVal context As
19 StreamingContext)

20 [JScript] protected function MethodAccessException(info : SerializationInfo,
21 context : StreamingContext);

22
23 *Description*

24 Initializes a new instance of the **System.MethodAccessException** class
25 with serialized data.

This constructor is called during deserialization to reconstitute the exception object transmitted over a stream. For more information, see . The object that holds the serialized object data. The contextual information about the source or destination.

MethodAccessException

Example Syntax:

ToString

[C#] public MethodAccessException(string message, Exception inner);

[C++] public: MethodAccessException(String* message, Exception* inner);

[VB] Public Sub New(ByVal message As String, ByVal inner As Exception)

[JScript] public function MethodAccessException(message : String, inner : Exception);

Description

Initializes a new instance of the **System.MethodAccessException** class with a specified error message and a reference to the inner exception that is the root cause of this exception.

When an **Exception** *X* is thrown as a direct result of a previous exception *Y*, the **System.Exception.InnerException** property of *X* should contain a reference to *Y*. The **InnerException** property returns the same value as was passed into the constructor, or **null** if the inner exception value was not supplied to the constructor. The error message that explains the reason for the exception. An instance of **System.Exception** that is the cause of the current **Exception**. If *inner* is non-null, then the current **Exception** is raised in a catch block handling *inner*.

HelpLink

HResult

InnerException

Message

Source

StackTrace

TargetSite

MissingFieldException class (System)

ToString

Description

The exception that is thrown when there is an attempt to dynamically access a field that does not exist.

System.MissingFieldException uses the HRESULT `COR_E_MISSINGFIELD`, which has the value 0x80131511.

MissingFieldException

Example Syntax:

ToString

System.MissingFieldException

Description

Initializes a new instance of the **System.MissingFieldException** class with default properties.

The following table shows the initial property values for an instance of

System.MissingFieldException .

MissingFieldException

Example Syntax:

ToString

[C#] public MissingFieldException(string message);

[C++] public: MissingFieldException(String* message);

[VB] Public Sub New(ByVal message As String)

[JScript] public function MissingFieldException(message : String);

Description

Initializes a new instance of the **System.MissingFieldException** class with a specified error message.

The following table shows the initial property values for an instance of **System.MissingFieldException** . The error message that explains the reason for the exception.

MissingFieldException

Example Syntax:

ToString

[C#] protected MissingFieldException(SerializationInfo info, StreamingContext context);

[C++] protected: MissingFieldException(SerializationInfo* info, StreamingContext context);

1 [VB] Protected Sub New(ByVal info As SerializationInfo, ByVal context As
2 StreamingContext)
3 [JScript] protected function MissingFieldException(info : SerializationInfo,
4 context : StreamingContext);

5
6 *Description*

7 Initializes a new instance of the **System.MissingFieldException** class with
8 serialized data.

9 This constructor is called during deserialization to reconstitute the
10 exception object transmitted over a stream. For more information, see . The object
11 that holds the serialized object data. The contextual information about the source
12 or destination.

13 MissingFieldException

14 *Example Syntax:*

15 ToString

16
17 [C#] public MissingFieldException(string message, Exception inner);

18 [C++] public: MissingFieldException(String* message, Exception* inner);

19 [VB] Public Sub New(ByVal message As String, ByVal inner As Exception)

20 [JScript] public function MissingFieldException(message : String, inner :
21 Exception);

22
23 *Description*

1 Initializes a new instance of the **System.MissingFieldException** class with
2 a specified error message and a reference to the inner exception that is the root
3 cause of this exception.

4 When an **Exception** *X* is thrown as a direct result of a previous exception *Y*,
5 the **System.Exception.InnerException** property of *X* should contain a reference
6 to *Y*. The **InnerException** property returns the same value as was passed into the
7 constructor, or **null** if the inner exception value was not supplied to the
8 constructor. The error message that explains the reason for the exception. An
9 instance of **System.Exception** that is the cause of the current **Exception**. If *inner*
10 is non-null, then the current **Exception** is raised in a catch block handling *inner*.

11 MissingFieldException

12 *Example Syntax:*

13 ToString

14
15 [C#] public MissingFieldException(string className, string fieldName);

16 [C++] public: MissingFieldException(String* className, String* fieldName);

17 [VB] Public Sub New(ByVal className As String, ByVal fieldName As String)

18 [JScript] public function MissingFieldException(className : String, fieldName :
19 String);

20
21 *Description*

22 Initializes a new instance of the **System.MissingFieldException** class with
23 the specified class name and field name. The name of the class in which access to
24 a nonexistent field was attempted. The name of the field that cannot be accessed.

25 HelpLink

1 HResult

2 InnerException

3 Message

4 ToString

5
6
7 *Description*

8 Gets the text string showing the signature of the missing field, the class
9 name, and the field name.

10 If the class name is not specified when the object is constructed, the default
11 text string inherited from the base class is returned. This property overrides
12 **System.MissingMemberException.Message** . The error message should be
13 localized.

14 Source

15 StackTrace

16 TargetSite

17 MissingMemberException class (System)

18 ToString

19
20
21 *Description*

22 The exception that is thrown when there is an attempt to dynamically
23 access a class member that does not exist.

24 Normally a compilation error is generated if the code attempts to access a
25 nonexistent member of a class.

1 ToString
2
3 [C#] protected string ClassName;
4 [C++] protected: String* ClassName;
5 [VB] Protected ClassName As String
6 [JScript] protected var ClassName : String;
7

8 *Description*

9 Holds the class name of the missing member.

10 ToString
11

12 [C#] protected string MemberName;
13 [C++] protected: String* MemberName;
14 [VB] Protected MemberName As String
15 [JScript] protected var MemberName : String;
16

17 *Description*

18 Holds the name of the missing member.

19 ToString
20

21 [C#] protected byte[] Signature;
22 [C++] protected: unsigned char Signature __gc[];
23 [VB] Protected Signature() As Byte
24 [JScript] protected var Signature : Byte[];
25

1
2 *Description*

3 Holds the signature of the missing member.

4 **System.MissingMemberException.Signature** contains a **System.Byte**
5 value that represents the signature of the missing member.

6 **MissingMemberException**

7 *Example Syntax:*

8 ToString

9
10 [C#] public MissingMemberException();

11 [C++] public: MissingMemberException();

12 [VB] Public Sub New()

13 [JScript] public function MissingMemberException(); Initializes a new instance of
14 the **System.MissingMemberException** class.

15
16 *Description*

17 Initializes a new instance of the **System.MissingMemberException** class
18 with default properties.

19 The following table shows the initial property values for an instance of
20 **System.MissingMemberException** .

21 **MissingMemberException**

22 *Example Syntax:*

23 ToString

24
25 [C#] public MissingMemberException(string message);

```

1 [C++] public: MissingMemberException(String* message);
2 [VB] Public Sub New(ByVal message As String)
3 [JScript] public function MissingMemberException(message : String);
4

```

Description

Initializes a new instance of the **System.MissingMemberException** class with a specified error message.

The following table shows the initial property values for an instance of **System.MissingMemberException** . The error message that explains the reason for the exception.

MissingMemberException

Example Syntax:

ToString

```

15 [C#] protected MissingMemberException(SerializationInfo info,
16 StreamingContext context);
17 [C++] protected: MissingMemberException(SerializationInfo* info,
18 StreamingContext context);
19 [VB] Protected Sub New(ByVal info As SerializationInfo, ByVal context As
20 StreamingContext)
21 [JScript] protected function MissingMemberException(info : SerializationInfo,
22 context : StreamingContext);
23

```

Description

1 Initializes a new instance of the **System.MissingMemberException** class
2 with serialized data.

3 This constructor is called during deserialization to reconstitute the
4 exception object transmitted over a stream. For more information, see . The object
5 that holds the serialized object data. The contextual information about the source
6 or destination.

7 **MissingMemberException**

8 *Example Syntax:*

9 **ToString**

10
11 [C#] public MissingMemberException(string message, Exception inner);
12 [C++] public: MissingMemberException(String* message, Exception* inner);
13 [VB] Public Sub New(ByVal message As String, ByVal inner As Exception)
14 [JScript] public function MissingMemberException(message : String, inner :
15 Exception);
16

17 *Description*

18 Initializes a new instance of the **System.MissingMemberException** class
19 with a specified error message and a reference to the inner exception that is the
20 root cause of this exception.

21 When an **Exception** *X* is thrown as a direct result of a previous exception *Y* ,
22 the **System.Exception.InnerException** property of *X* should contain a reference
23 to *Y* . The **InnerException** property returns the same value as was passed into the
24 constructor, or **null** if the inner exception value was not supplied to the
25 constructor. The error message that explains the reason for the exception. An

instance of **System.Exception** that is the cause of the current **Exception**. If *inner* is non-null, then the current **Exception** is raised in a catch block handling *inner* .

MissingMemberException

Example Syntax:

ToString

```
[C#] public MissingMemberException(string className, string memberName);
```

```
[C++] public: MissingMemberException(String* className, String*  
memberName);
```

```
[VB] Public Sub New(ByVal className As String, ByVal memberName As  
String)
```

```
[JScript] public function MissingMemberException(className : String,  
memberName : String);
```

Description

Initializes a new instance of the **System.MissingMemberException** class with the specified class name and member name. The name of the class in which access to a nonexistent member was attempted. The name of the member that cannot be accessed.

HelpLink

HResult

InnerException

Message

ToString

1
2
3 *Description*

4 Gets the text string showing the class name, the member name, and the
5 signature of the missing member. If the class name is not specified when the
6 object is constructed, the default text string inherited from the base class is
7 returned.

8 This property overrides **System.Exception.Message** . The error message
9 should be localized.

10 Source

11 StackTrace

12 TargetSite

13 GetObjectData
14

15 [C#] public override void GetObjectData(SerializationInfo info, StreamingContext
16 context);

17 [C++] public: void GetObjectData(SerializationInfo* info, StreamingContext
18 context);

19 [VB] Overrides Public Sub GetObjectData(ByVal info As SerializationInfo,
20 ByVal context As StreamingContext)

21 [JScript] public override function GetObjectData(info : SerializationInfo, context :
22 StreamingContext);
23

24 *Description*
25

1 Sets the **System.Runtime.Serialization.SerializationInfo** object with the
2 class name, the member name, the signature of the missing member, and
3 additional exception information.

4 **System.TypeLoadException.GetObjectData(System.Runtime.Serializat**
5 **ion.SerializationInfo, System.Runtime.Serialization.StreamingContext)** sets a
6 **System.Runtime.Serialization.SerializationInfo** with all the exception object
7 data targeted for serialization. During deserialization, the exception object is
8 reconstituted from the **System.Runtime.Serialization.SerializationInfo**
9 transmitted over the stream. The object that holds the serialized object data. The
10 contextual information about the source or destination.

11 MissingMethodException class (System)

12 ToString

13
14
15 *Description*

16 The exception that is thrown when there is an attempt to dynamically
17 access a method that does not exist.

18 **System.MissingMethodException** uses the HRESULT
19 COR_E_MISSINGMETHOD, which has the value 0x80131513.

20 MissingMethodException

21 *Example Syntax:*

22 ToString

23 **System.MissingMethodException**

24
25 *Description*

1 Initializes a new instance of the **System.MissingMethodException** class
2 with default properties.

3 The following table shows the initial property values for an instance of
4 **System.MissingMethodException** .

5 MissingMethodException

6 *Example Syntax:*

7 ToString

9 [C#] public MissingMethodException(string message);

10 [C++] public: MissingMethodException(String* message);

11 [VB] Public Sub New(ByVal message As String)

12 [JScript] public function MissingMethodException(message : String);

14 *Description*

15 Initializes a new instance of the **System.MissingMethodException** class
16 with a specified error message.

17 The following table shows the initial property values for an instance of
18 **System.MissingMethodException** . The error message that explains the reason
19 for the exception.

20 MissingMethodException

21 *Example Syntax:*

22 ToString

24 [C#] protected MissingMethodException(SerializationInfo info, StreamingContext
25 context);

1 [C++] protected: MissingMethodException(SerializationInfo* info,
 2 StreamingContext context);
 3 [VB] Protected Sub New(ByVal info As SerializationInfo, ByVal context As
 4 StreamingContext)
 5 [JScript] protected function MissingMethodException(info : SerializationInfo,
 6 context : StreamingContext);
 7

8 *Description*

9 Initializes a new instance of the **System.MissingMethodException** class
 10 with serialized data.

11 This constructor is called during deserialization to reconstitute the
 12 exception object transmitted over a stream. For more information, see . The object
 13 that holds the serialized object data. The contextual information about the source
 14 or destination.

15 MissingMethodException

16 *Example Syntax:*

17 ToString

18
 19 [C#] public MissingMethodException(string message, Exception inner);
 20 [C++] public: MissingMethodException(String* message, Exception* inner);
 21 [VB] Public Sub New(ByVal message As String, ByVal inner As Exception)
 22 [JScript] public function MissingMethodException(message : String, inner :
 23 Exception);
 24

25 *Description*

1 Initializes a new instance of the **System.MissingMethodException** class
2 with a specified error message and a reference to the inner exception that is the
3 root cause of this exception.

4 When an **Exception** *X* is thrown as a direct result of a previous exception *Y*,
5 the **System.Exception.InnerException** property of *X* should contain a reference
6 to *Y*. The **InnerException** property returns the same value as was passed into the
7 constructor, or **null** if the inner exception value was not supplied to the
8 constructor. The error message that explains the reason for the exception. An
9 instance of **System.Exception** that is the cause of the current **Exception**. If *inner*
10 is non-null, then the current **Exception** is raised in a catch block handling *inner*.

11 MissingMethodException

12 *Example Syntax:*

13 ToString

14
15 [C#] public MissingMethodException(string className, string methodName);

16 [C++] public: MissingMethodException(String* className, String*
17 methodName);

18 [VB] Public Sub New(ByVal className As String, ByVal methodName As
19 String)

20 [JScript] public function MissingMethodException(className : String,
21 methodName : String);

22
23 *Description*

24 Initializes a new instance of the **System.MissingMethodException** class
25 with the specified class name and method name. The name of the class in which

access to a nonexistent method was attempted. The name of the method that cannot be accessed.

HelpLink

HResult

InnerException

Message

ToString

Description

Gets the text string showing the class name, the method name, and the signature of the missing method.

If the class name is not specified when the object is constructed, the default text string inherited from the base class is returned.

Source

StackTrace

TargetSite

MTAThreadAttribute class (System)

ToString

Description

Indicates the default threading model for an application is multi-threaded apartment.

Only apply this attribute to the main method of an application.

MTAThreadAttribute

Example Syntax:

ToString

[C#] public MTAThreadAttribute();

[C++] public: MTAThreadAttribute();

[VB] Public Sub New()

[JScript] public function MTAThreadAttribute();

Description

Initializes a new instance of the **System.MTAThreadAttribute** class.

TypeId

MulticastDelegate class (System)

ToString

Description

Represents a multicast delegate; that is, a delegate that can have more than one element in its invocation list.

All multicast delegates are derived from class **MulticastDelegate** .

MulticastDelegate

Example Syntax:

ToString

[C#] protected MulticastDelegate(object target, string method);

1 [C++] protected: MulticastDelegate(Object* target, String* method);

2 [VB] Protected Sub New(ByVal target As Object, ByVal method As String)

3 [JScript] protected function MulticastDelegate(target : Object, method : String);

4 Initializes a new instance of the **MulticastDelegate** class.

5
6 *Description*

7 Initializes a new instance of the **MulticastDelegate** class. This constructor
8 is called from the class generated by the compiler-generated code.

9 This must match the constructor in **Delegate** . The object on which the
10 specified method is defined. The name of the method for which to create a
11 delegate.

12 MulticastDelegate

13 *Example Syntax:*

14 ToString

15
16 [C#] protected MulticastDelegate(Type target, string method);

17 [C++] protected: MulticastDelegate(Type* target, String* method);

18 [VB] Protected Sub New(ByVal target As Type, ByVal method As String)

19 [JScript] protected function MulticastDelegate(target : Type, method : String);

20
21 *Description*

22 Initializes a new instance of the **MulticastDelegate** class. This constructor
23 is called from a class to generate a delegate based upon a static method name and
24 the **Type** object for the class defining the method. The **Type** object that represents
25

1 the class that the specified method is defined on. The name of the static method for
2 which to create a delegate.

3 Method

4 Target

5 CombineImpl

6
7 [C#] protected override Delegate CombineImpl(Delegate follow);

8 [C++] protected: Delegate* CombineImpl(Delegate* follow);

9 [VB] Overrides Protected Function CombineImpl(ByVal follow As Delegate) As

10 Delegate

11 [JScript] protected override function CombineImpl(follow : Delegate) : Delegate;

12
13 *Description*

14 Combines this **System.Delegate** with the passed **Delegate** to form a new
15 delegate.

16 *Return Value:* A **Delegate** object as the new root. The Delegate with which to
17 combine this Delegate.

18 DynamicInvokeImpl

19
20 [C#] protected override object DynamicInvokeImpl(object[] args);

21 [C++] protected: Object* DynamicInvokeImpl(Object* args __gc[]);

22 [VB] Overrides Protected Function DynamicInvokeImpl(ByVal args() As Object)

23 As Object

24 [JScript] protected override function DynamicInvokeImpl(args : Object[]) :

25 Object;

Description

Processes the full invocation list.

Return Value: An array of type **Object** that contains the return value of the encapsulated method. The arguments to be passed to the encapsulated method.

Equals

[C#] public override bool Equals(object obj);

[C++] public: bool Equals(Object* obj);

[VB] Overrides Public Function Equals(ByVal obj As Object) As Boolean

[JScript] public override function Equals(obj : Object) : Boolean;

Description

Determines whether this multicast delegate and the specified object are equal.

Return Value: **true** if *obj* and this instance have the same invocation lists; otherwise, **false** .

Two delegates, whether single- or multi-cast, are equal if they have the same invocation lists. Two invocation lists are considered identical if they have the same order, and the corresponding elements from the two lists represent the same method and target. The object to compare with this instance.

GetHashCode

[C#] public override int GetHashCode();

[C++] public: int GetHashCode();

1 [VB] Overrides Public Function GetHashCode() As Integer

2 [JScript] public override function GetHashCode() : int;

3
4 *Description*

5 Returns the hash code for this instance.

6 *Return Value:* A 32-bit signed integer hash code.

7 *GetInvocationList*

8
9 [C#] public override Delegate[] GetInvocationList();

10 [C++] public: Delegate* GetInvocationList() [];

11 [VB] Overrides Public Function GetInvocationList() As Delegate()

12 [JScript] public override function GetInvocationList() : Delegate[];

13
14 *Description*

15 Returns the invocation list of this multicast delegate, in invocation order.

16 *Return Value:* An array of delegates in the invocation list.

17 *op_Equality*

18
19 [C#] public static new bool operator ==(MulticastDelegate d1, MulticastDelegate
20 d2);

21 [C++] public: static bool op_Equality(MulticastDelegate* d1, MulticastDelegate*
22 d2);

23 [VB] returnValue = MulticastDelegate.op_Equality(d1, d2)

24 [JScript] returnValue = d1 == d2;

Description

Determines whether two MulticastDelegate objects are equal.

Return Value: True if d1 and d2 have the same invocation lists; otherwise false.

Two delegates, whether single- or multi-cast, are equal if they have the same invocation lists. Two invocation lists are considered identical if they have the same order, and the corresponding elements from the two lists represent the same method and target. The left operand. The right operand.

op_Inequality

```
[C#] public static new bool operator !=(MulticastDelegate d1, MulticastDelegate d2);
```

```
[C++] public: static bool op_Inequality(MulticastDelegate* d1, MulticastDelegate* d2);
```

```
[VB] returnValue = MulticastDelegate.op_Inequality(d1, d2)
```

```
[JScript] returnValue = d1 != d2;
```

Description

Determines whether two MulticastDelegate objects are not equal.

Return Value: True if d1 and d2 do not have the same invocation lists; otherwise false.

Two delegates, whether single- or multi-cast, are equal if they have the same invocation lists. Two invocation lists are considered identical if they have the same order, and the corresponding elements from the two lists represent the same method and target. The left operand. The right operand.

RemoveImpl

[C#] protected override Delegate RemoveImpl(Delegate value);

[C++] protected: Delegate* RemoveImpl(Delegate* value);

[VB] Overrides Protected Function RemoveImpl(ByVal value As Delegate) As Delegate

[JScript] protected override function RemoveImpl(value : Delegate) : Delegate;

Description

Searches the invocation list for an element that has **System.Delegate** -based equality with *value* .

Return Value: A new **Delegate** if an element on the invocation list is found that has **Delegate** -based equality with *value* (and thus is removed from the invocation list). If such an element is not found, the current invocation list is returned. The **Delegate** to search for in the invocation list.

MulticastNotSupportedException class (System)

ToString

Description

The exception that is thrown when there is an attempt to combine two instances of a non-combinable delegate type unless one of the operands is **null** . This class cannot be inherited.

A valid delegate combination is made when one or both operands is a combinable delegate type. If both operands are non-combinable delegate type,

then one operand must be **null** . A combinable delegate type must satisfy the following conditions: The declared return type of the delegate must be void.

MulticastNotSupportedException

Example Syntax:

ToString

[C#] public MulticastNotSupportedException();

[C++] public: MulticastNotSupportedException();

[VB] Public Sub New()

[JScript] public function MulticastNotSupportedException(); Initializes a new instance of the **System.MulticastNotSupportedException** class.

Description

Initializes a new instance of the **System.MulticastNotSupportedException** class with default properties.

The following table shows the initial property values for an instance of **System.MulticastNotSupportedException** .

MulticastNotSupportedException

Example Syntax:

ToString

[C#] public MulticastNotSupportedException(string message);

[C++] public: MulticastNotSupportedException(String* message);

[VB] Public Sub New(ByVal message As String)

[JScript] public function MulticastNotSupportedException(message : String);

Description

Initializes a new instance of the **System.MulticastNotSupportedException** class with a specified error message.

The following table shows the initial property values for an instance of **System.MulticastNotSupportedException**. The error message that explains the reason for the exception.

MulticastNotSupportedException

Example Syntax:

ToString

```
[C#] public MulticastNotSupportedException(string message, Exception inner);
```

```
[C++] public: MulticastNotSupportedException(String* message, Exception* inner);
```

```
[VB] Public Sub New(ByVal message As String, ByVal inner As Exception)
```

```
[JScript] public function MulticastNotSupportedException(message : String, inner : Exception);
```

Description

Initializes a new instance of the **System.MulticastNotSupportedException** class with a specified error message and a reference to the inner exception that is the root cause of this exception.

When an **Exception** *X* is thrown as a direct result of a previous exception *Y*, the **System.Exception.InnerException** property of *X* should contain a reference to *Y*. The **InnerException** property returns the same value as was passed into the

1 constructor, or **null** if the inner exception value was not supplied to the
2 constructor. The error message that explains the reason for the exception. An
3 instance of **System.Exception** that is the cause of the current **Exception**. If *inner*
4 is non-null, then the current **Exception** is raised in a catch block handling *inner* .

5 HelpLink

6 HResult

7 InnerException

8 Message

9 Source

10 StackTrace

11 TargetSite

12 NonSerializedAttribute class (System)

13 ToString

16 *Description*

17 Indicates that a field of a serializable class should not be serialized. This
18 class cannot be inherited.

19 The target objects for the **System.NonSerializedAttribute** are public and
20 private fields of a serializable class. By default, classes are not serializable unless
21 they are marked with the **System.SerializableAttribute** . During the serialization
22 process all the public and private fields of a class are serialized by default. Fields
23 that must not be serialized can be marked with the
24 **System.NonSerializedAttribute** , which instructs the serialization process to
25 ignore the target field during serialization.

NonSerializedAttribute

Example Syntax:

ToString

[C#] public NonSerializedAttribute();

[C++] public: NonSerializedAttribute();

[VB] Public Sub New()

[JScript] public function NonSerializedAttribute();

Description

Initializes a new instance of the **System.NonSerializedAttribute** class.

TypeId

NotFiniteNumberException class (System)

ToString

Description

The exception that is thrown when a floating-point value is positive infinity, negative infinity, or Not-a-Number (NaN).

Applications written in C# will not throw this exception.

NotFiniteNumberException

Example Syntax:

ToString

[C#] public NotFiniteNumberException();

[C++] public: NotFiniteNumberException();

[VB] Public Sub New()

[JScript] public function NotFiniteNumberException(); Initializes a new instance of the **System.NotFiniteNumberException** class.

Description

Initializes a new instance of the **System.NotFiniteNumberException** class with default properties.

The following table shows the initial property values for an instance of **System.NotFiniteNumberException**.

NotFiniteNumberException

Example Syntax:

ToString

[C#] public NotFiniteNumberException(double offendingNumber);

[C++] public: NotFiniteNumberException(double offendingNumber);

[VB] Public Sub New(ByVal offendingNumber As Double)

[JScript] public function NotFiniteNumberException(offendingNumber : double);

Description

Initializes a new instance of the **System.NotFiniteNumberException** class with the invalid number.

The *offendingNumber* parameter must be both a **System.Double** and an invalid number. The invalid number.

NotFiniteNumberException

Example Syntax:

ToString

[C#] public NotFiniteNumberException(string message);

[C++] public: NotFiniteNumberException(String* message);

[VB] Public Sub New(ByVal message As String)

[JScript] public function NotFiniteNumberException(message : String);

Description

Initializes a new instance of the **System.NotFiniteNumberException** class with a specified error message.

The following table shows the initial property values for an instance of **System.NotFiniteNumberException** . The error message that explains the reason for the exception.

NotFiniteNumberException

Example Syntax:

ToString

[C#] protected NotFiniteNumberException(SerializationInfo info, StreamingContext context);

[C++] protected: NotFiniteNumberException(SerializationInfo* info, StreamingContext context);

[VB] Protected Sub New(ByVal info As SerializationInfo, ByVal context As StreamingContext)

[JScript] protected function NotFiniteNumberException(info : SerializationInfo,

context : StreamingContext);

Description

Initializes a new instance of the **System.NotFiniteNumberException** class with serialized data.

This constructor is called during deserialization to reconstitute the exception object transmitted over a stream. For more information, see . The object that holds the serialized object data. The contextual information about the source or destination.

NotFiniteNumberException

Example Syntax:

ToString

[C#] public NotFiniteNumberException(string message, double offendingNumber);

[C++] public: NotFiniteNumberException(String* message, double offendingNumber);

[VB] Public Sub New(ByVal message As String, ByVal offendingNumber As Double)

[JScript] public function NotFiniteNumberException(message : String, offendingNumber : double);

Description

Initializes a new instance of the **System.NotFiniteNumberException** class with a specified error message and the invalid number.

The following table shows the initial property values for an instance of **System.NotFiniteNumberException**. The error message that explains the reason for the exception. The invalid number.

NotFiniteNumberException

Example Syntax:

ToString

[C#] public NotFiniteNumberException(string message, double offendingNumber, Exception innerException);

[C++] public: NotFiniteNumberException(String* message, double offendingNumber, Exception* innerException);

[VB] Public Sub New(ByVal message As String, ByVal offendingNumber As Double, ByVal innerException As Exception)

[JScript] public function NotFiniteNumberException(message : String, offendingNumber : double, innerException : Exception);

Description

Initializes a new instance of the **System.NotFiniteNumberException** class with a specified error message, the invalid number, and a reference to the inner exception that is the root cause of this exception.

When an **Exception** *X* is thrown as a direct result of a previous exception *Y*, the **System.Exception.InnerException** property of *X* should contain a reference to *Y*. The **InnerException** property returns the same value as was passed into the constructor, or **null** if the inner exception value was not supplied to the constructor. The error message that explains the reason for the exception. The

1 invalid number. An instance of **System.Exception** that is the cause of the current
2 **Exception**. If *innerException* is non-null, then the current **Exception** is raised in a
3 catch block handling *innerException* .

4 HelpLink

5 HResult

6 InnerException

7 Message

8 OffendingNumber

9 ToString

10
11
12 *Description*

13 Gets the invalid number that is a positive infinity, a negative infinity, or
14 Not-a-Number (NaN).

15 Source

16 StackTrace

17 TargetSite

18 GetObjectData

19
20 [C#] public override void GetObjectData(SerializationInfo info, StreamingContext
21 context);

22 [C++] public: void GetObjectData(SerializationInfo* info, StreamingContext
23 context);

24 [VB] Overrides Public Sub GetObjectData(ByVal info As SerializationInfo,
25 ByVal context As StreamingContext)

1 [JScript] public override function GetObjectData(info : SerializationInfo, context :
2 StreamingContext);

4 *Description*

5 Sets the **System.Runtime.Serialization.SerializationInfo** object with the
6 invalid number and additional exception information.

7 **System.TypeLoadException.GetObjectData(System.Runtime.Serializat**
8 **ion.SerializationInfo, System.Runtime.Serialization.StreamingContext)** sets a
9 **System.Runtime.Serialization.SerializationInfo** with all the exception object
10 data targeted for serialization. During deserialization, the exception object is
11 reconstituted from the **System.Runtime.Serialization.SerializationInfo**
12 transmitted over the stream. The object that holds the serialized object data. The
13 contextual information about the source or destination.

14 NotImplementedException class (System)

15 ToString

18 *Description*

19 The exception that is thrown when a requested method or operation is not
20 implemented.

21 **System.NotImplementedException** uses the default
22 **System.Object.Equals(System.Object)** implementation, which supports
23 reference equality. For a list of initial values for an instance of
24 **System.NotImplementedException**, see the
25 **System.NotImplementedException.#ctor** constructors.

NotImplementedException

Example Syntax:

ToString

[C#] public NotImplementedException();

[C++] public: NotImplementedException();

[VB] Public Sub New()

[JScript] public function NotImplementedException(); Initializes a new instance of the **System.NotImplementedException** class.

Description

Initializes a new instance of the **System.NotImplementedException** class with default properties.

The following table shows the initial property values for an instance of **System.NotImplementedException**.

NotImplementedException

Example Syntax:

ToString

[C#] public NotImplementedException(string message);

[C++] public: NotImplementedException(String* message);

[VB] Public Sub New(ByVal message As String)

[JScript] public function NotImplementedException(message : String);

Description

1 Initializes a new instance of the **System.NotImplementedException** class
2 with a specified error message.

3 The following table shows the initial property values for an instance of
4 **System.NotImplementedException** . The error message that explains the reason
5 for the exception.

6 NotImplementedException

7 *Example Syntax:*

8 ToString

9
10 [C#] protected NotImplementedException(SerializationInfo info,
11 StreamingContext context);

12 [C++] protected: NotImplementedException(SerializationInfo* info,
13 StreamingContext context);

14 [VB] Protected Sub New(ByVal info As SerializationInfo, ByVal context As
15 StreamingContext)

16 [JScript] protected function NotImplementedException(info : SerializationInfo,
17 context : StreamingContext);

18
19 *Description*

20 Initializes a new instance of the **System.NotImplementedException** class
21 with serialized data. The object that holds the serialized object data. The
22 contextual information about the source or destination.

23 NotImplementedException

24 *Example Syntax:*

25 ToString

```

1
2 [C#] public NotImplementedException(string message, Exception inner);
3 [C++] public: NotImplementedException(String* message, Exception* inner);
4 [VB] Public Sub New(ByVal message As String, ByVal inner As Exception)
5 [JScript] public function NotImplementedException(message : String, inner :
6 Exception);
7

```

8 *Description*

9 Initializes a new instance of the **System.NotImplementedException** class
10 with a specified error message and a reference to the inner exception that is the
11 root cause of this exception.

12 When an **Exception** *X* is thrown as a direct result of a previous exception *Y* ,
13 the **System.Exception.InnerException** property of *X* should contain a reference
14 to *Y* . The **InnerException** property returns the same value as was passed into the
15 constructor, or **null** if the inner exception value was not supplied to the
16 constructor. The error message that explains the reason for the exception. An
17 instance of **System.Exception** that is the cause of the current **Exception**. If *inner*
18 is non-null, then the current **Exception** is raised in a catch block handling *inner* .

19 HelpLink

20 HResult

21 InnerException

22 Message

23 Source

24 StackTrace

25 TargetSite

1 NotSupportedException class (System)

2 ToString

3
4
5 *Description*

6 The exception that is thrown when an invoked method is not supported, or
7 when there is an attempt to read, seek, or write to a stream that does not support
8 the invoked functionality.

9 There are methods that are not supported in the base class, with the
10 expectation that these methods will be implemented in the derived classes instead.
11 The derived class might implement only a subset of the methods from the base
12 class, and throw **System.NotSupportedException** for the unsupported methods.

13 NotSupportedException

14 *Example Syntax:*

15 ToString

16
17 [C#] public NotSupportedException();

18 [C++] public: NotSupportedException();

19 [VB] Public Sub New()

20 [JScript] public function NotSupportedException(); Initializes a new instance of
21 the **System.NotSupportedException** class.

22
23 *Description*

24 Initializes a new instance of the **System.NotSupportedException** class
25 with default properties.

The following table shows the initial property values for an instance of **System.NotSupportedException**.

NotSupportedException

Example Syntax:

ToString

[C#] public NotSupportedException(string message);

[C++] public: NotSupportedException(String* message);

[VB] Public Sub New(ByVal message As String)

[JScript] public function NotSupportedException(message : String);

Description

Initializes a new instance of the **System.NotSupportedException** class with a specified error message.

The following table shows the initial property values for an instance of **System.NotSupportedException**. The error message that explains the reason for the exception.

NotSupportedException

Example Syntax:

ToString

[C#] protected NotSupportedException(SerializationInfo info, StreamingContext context);

[C++] protected: NotSupportedException(SerializationInfo* info, StreamingContext context);

[VB] Protected Sub New(ByVal info As SerializationInfo, ByVal context As StreamingContext)

[JScript] protected function NotSupportedException(info : SerializationInfo, context : StreamingContext);

Description

Initializes a new instance of the **System.NotSupportedException** class with serialized data.

This constructor is called during deserialization to reconstitute the exception object transmitted over a stream. For more information, see . The object that holds the serialized object data. The contextual information about the source or destination.

NotSupportedException

Example Syntax:

ToString

[C#] public NotSupportedException(string message, Exception innerException);

[C++] public: NotSupportedException(String* message, Exception* innerException);

[VB] Public Sub New(ByVal message As String, ByVal innerException As Exception)

[JScript] public function NotSupportedException(message : String, innerException : Exception);

Description

1 Initializes a new instance of the **System.NotSupportedException** class
2 with a specified error message and a reference to the inner exception that is the
3 root cause of this exception.

4 When an **Exception** *X* is thrown as a direct result of a previous exception *Y* ,
5 the **System.Exception.InnerException** property of *X* should contain a reference
6 to *Y* . The **InnerException** property returns the same value as was passed into the
7 constructor, or **null** if the inner exception value was not supplied to the
8 constructor. The error message that explains the reason for the exception. An
9 instance of **System.Exception** that is the cause of the current **Exception**. If
10 *innerException* is non-null, then the current **Exception** is raised in a catch block
11 handling *innerException* .

12 HelpLink

13 HResult

14 InnerException

15 Message

16 Source

17 StackTrace

18 TargetSite

19 NullReferenceException class (System)

20 ToString

21
22
23 *Description*

24 The exception that is thrown when there is an attempt to dereference a null
25 object reference.

System.NullReferenceException uses the HRESULT
COR_E_NULLREFERENCE, which has the value 0x80004003.

NullReferenceException

Example Syntax:

ToString

[C#] public NullReferenceException();

[C++] public: NullReferenceException();

[VB] Public Sub New()

[JScript] public function NullReferenceException(); Initializes a new instance of
the **System.NullReferenceException** class.

Description

Initializes a new instance of the **System.NullReferenceException** class
with default properties.

The following table shows the initial property values for an instance of
System.NullReferenceException.

NullReferenceException

Example Syntax:

ToString

[C#] public NullReferenceException(string message);

[C++] public: NullReferenceException(String* message);

[VB] Public Sub New(ByVal message As String)

[JScript] public function NullReferenceException(message : String);

1
2 *Description*

3 Initializes a new instance of the **System.NullReferenceException** class
4 with a specified error message.

5 The following table shows the initial property values for an instance of
6 **System.NullReferenceException** . The error message that explains the reason for
7 the exception.

8 NullReferenceException

9 *Example Syntax:*

10 ToString

11
12 [C#] protected NullReferenceException(SerializationInfo info, StreamingContext
13 context);

14 [C++] protected: NullReferenceException(SerializationInfo* info,
15 StreamingContext context);

16 [VB] Protected Sub New(ByVal info As SerializationInfo, ByVal context As
17 StreamingContext)

18 [JScript] protected function NullReferenceException(info : SerializationInfo,
19 context : StreamingContext);

20
21 *Description*

22 Initializes a new instance of the **System.NullReferenceException** class
23 with serialized data.

24 This constructor is called during deserialization to reconstitute the
25 exception object transmitted over a stream. For more information, see . The object

that holds the serialized object data. The contextual information about the source or destination.

NullReferenceException

Example Syntax:

ToString

```
[C#] public NullReferenceException(string message, Exception innerException);
```

```
[C++] public: NullReferenceException(String* message, Exception*  
innerException);
```

```
[VB] Public Sub New(ByVal message As String, ByVal innerException As  
Exception)
```

```
[JScript] public function NullReferenceException(message : String,  
innerException : Exception);
```

Description

Initializes a new instance of the **System.NullReferenceException** class with a specified error message and a reference to the inner exception that is the root cause of this exception.

When an **Exception** *X* is thrown as a direct result of a previous exception *Y*, the **System.Exception.InnerException** property of *X* should contain a reference to *Y*. The **InnerException** property returns the same value as was passed into the constructor, or **null** if the inner exception value was not supplied to the constructor. The error message that explains the reason for the exception. An instance of **System.Exception** that is the cause of the current **Exception**. If

1 *innerException* is non-null, then the current **Exception** is raised in a catch block
2 handling *innerException* .

3 HelpLink

4 HResult

5 InnerException

6 Message

7 Source

8 StackTrace

9 TargetSite

10 Object class (System)

11 ToString

12
13
14 *Description*

15 Supports all classes in the .NET Framework class hierarchy and provides
16 low-level services to derived classes. This is the ultimate superclass of all classes
17 in the .NET Framework; it is the root of the type hierarchy.

18 Languages typically do not require a class to declare inheritance from
19 **System.Object** because the inheritance is implicit.

20 Object

21 *Example Syntax:*

22 ToString

23
24 [C#] public Object();

25 [C++] public: Object();

1 [VB] Public Sub New()

2 [JScript] public function Object();

3
4 *Description*

5 Initializes a new instance of the **System.Object** class.

6 This constructor is called by constructors in derived classes, but it can also
7 be used to directly create an instance of the **System.Object** class.

8 Equals

9
10 [C#] public virtual bool Equals(object obj);

11 [C++] public: virtual bool Equals(Object* obj);

12 [VB] Overridable Public Function Equals(ByVal obj As Object) As Boolean

13 [JScript] public function Equals(obj : Object) : Boolean; Determines whether two
14 **System.Object** instances are equal.

15
16 *Description*

17 Determines whether the specified **System.Object** is equal to the current
18 **System.Object** .

19 *Return Value:* **true** if the specified **System.Object** is equal to the current
20 **System.Object** ; otherwise, **false** .

21 The default implementation of **System.Object.Equals(System.Object)**
22 supports reference equality only, but derived classes can override this method to
23 support value equality. The **System.Object** to compare with the current
24 **System.Object**.

25 Equals

1
2 [C#] public static bool Equals(object objA, object objB);

3 [C++] public: static bool Equals(Object* objA, Object* objB);

4 [VB] Public Shared Function Equals(ByVal objA As Object, ByVal objB As
5 Object) As Boolean

6 [JScript] public static function Equals(objA : Object, objB : Object) : Boolean;

7
8 *Description*

9 Determines whether the specified **System.Object** instances are considered
10 equal.

11 *Return Value:* **true** if *objA* is the same instance as *objB* or if both are null
12 references or if objA.Equals(objB) returns **true** ; otherwise, **false** .

13 The default implementation of **System.Object.Equals(System.Object)**
14 supports reference equality only, but derived classes can override this method to
15 support value equality. The first **System.Object** to compare. The second
16 **System.Object** to compare.

17 *Finalize*

18
19 [C#] ~Object();

20 [C++] ~Object();

21 [VB] Overrides Protected Sub Finalize()

22 [JScript] protected override function Finalize();

23
24 *Description*

1 Allows an **System.Object** to attempt to free resources and perform other
2 cleanup operations before the **System.Object** is reclaimed by garbage collection.

3 **System.Object.Finalize** is protected and, therefore, is accessible only
4 through this class or a derived class.

5 GetHashCode

6
7 [C#] public virtual int GetHashCode();

8 [C++] public: virtual int GetHashCode();

9 [VB] Overridable Public Function GetHashCode() As Integer

10 [JScript] public function GetHashCode() : int;

11 12 *Description*

13 Serves as a hash function for a particular type, suitable for use in hashing
14 algorithms and data structures like a hash table.

15 *Return Value:* A hash code for the current **System.Object** .

16 This method can be overridden by a derived class. Value classes must
17 override this method to provide a hash function that is appropriate for the class and
18 that ensures a better distribution in the hash table. Classes that might be used as a
19 key in a hash table must also override this method, because objects that are used as
20 keys in a hash table are required to generate their own hash code through this
21 method.

22 GetType

23
24 [C#] public Type GetType();

25 [C++] public: Type* GetType();

1 [VB] Public Function GetType() As Type

2 [JScript] public function GetType() : Type;

3
4 *Description*

5 Gets the **System.Type** of the current instance.

6 *Return Value:* The **System.Type** instance that represents the exact runtime type of
7 the current instance.

8 For two objects x and y that have identical runtime types,
9 `Object.ReferenceEquals(x.GetType(),y.GetType())` returns **true** .

10 **MemberwiseClone**

11
12 [C#] protected object MemberwiseClone();

13 [C++] protected: Object* MemberwiseClone();

14 [VB] Protected Function MemberwiseClone() As Object

15 [JScript] protected function MemberwiseClone() : Object;

16
17 *Description*

18 Creates a shallow copy of the current **System.Object** .

19 *Return Value:* A shallow copy of the current **System.Object** .

20 This method cannot be overridden; a derived class should implement the
21 **System.ICloneable** interface if a shallow copy is not appropriate.

22 **ReferenceEquals**

23
24 [C#] public static bool ReferenceEquals(object objA, object objB);

25 [C++] public: static bool ReferenceEquals(Object* objA, Object* objB);

1 [VB] Public Shared Function ReferenceEquals(ByVal objA As Object, ByVal
2 objB As Object) As Boolean

3 [JScript] public static function ReferenceEquals(objA : Object, objB : Object) :
4 Boolean;

5
6 *Description*

7 Determines whether the specified **System.Object** instances are the same
8 instance.

9 *Return Value:* **true** if *objA* is the same instance as *objB* or if both are null
10 references; otherwise, **false** . The first **System.Object** to compare. The second
11 **System.Object** to compare.

12 ToString

13
14 [C#] public virtual string ToString();

15 [C++] public: virtual String* ToString();

16 [VB] Overridable Public Function ToString() As String

17 [JScript] public function ToString() : String;

18
19 *Description*

20 Returns a **System.String** that represents the current **System.Object** .

21 *Return Value:* A **System.String** that represents the current **System.Object** .

22 This method returns a human-readable string that is culture-sensitive. For
23 example, for an instance of the **System.Double** class whose value is zero, the
24 implementation of **System.Double.ToString** might return "0.00" or "0,00"
25 depending on the current UI culture.

MSI-862US.APP

1	ObjectDisposedException class (System)
2	ToString
3	ObjectDisposedException
4	<i>Example Syntax:</i>
5	ToString
6	System.ObjectDisposedException
7	ObjectDisposedException
8	<i>Example Syntax:</i>
9	ToString
10	ObjectDisposedException
11	<i>Example Syntax:</i>
12	ToString
13	System.ObjectDisposedException
14	HelpLink
15	HResult
16	InnerException
17	Message
18	ToString
19	ObjectName
20	ToString
21	Source
22	StackTrace
23	TargetSite
24	ObsoleteAttribute class (System)
25	ToString

Description

Marks the program elements that are no longer in use. This class cannot be inherited.

System.ObsoleteAttribute is applicable to all program elements except assemblies, modules, parameters or return values. Marking an element as obsolete informs the users that the element will be removed in future versions of the product or that the functionality provided by the element is made internal to your application.

ObsoleteAttribute

Example Syntax:

ToString

[C#] public ObsoleteAttribute();

[C++] public: ObsoleteAttribute();

[VB] Public Sub New()

[JScript] public function ObsoleteAttribute(); Initializes a new instance of the **System.ObsoleteAttribute** class.

Description

Initializes a new instance of the **System.ObsoleteAttribute** class with default properties.

The following table shows the initial property values for an instance of **System.ObsoleteAttribute**.

ObsoleteAttribute

Example Syntax:

ToString

[C#] public ObsoleteAttribute(string message);

[C++] public: ObsoleteAttribute(String* message);

[VB] Public Sub New(ByVal message As String)

[JScript] public function ObsoleteAttribute(message : String);

Description

Initializes a new instance of the **System.ObsoleteAttribute** class with a specified workaround message.

The following table shows the initial property values for an instance of **System.ObsoleteAttribute** . The text string that describes alternative workarounds.

ObsoleteAttribute

Example Syntax:

ToString

[C#] public ObsoleteAttribute(string message, bool error);

[C++] public: ObsoleteAttribute(String* message, bool error);

[VB] Public Sub New(ByVal message As String, ByVal error As Boolean)

[JScript] public function ObsoleteAttribute(message : String, error : Boolean);

Description

1 Initializes a new instance of the **System.ObsoleteAttribute** class with a
2 workaround message and a Boolean value indicating whether the obsolete element
3 usage is considered an error.

4 The following table shows the initial property values for an instance of
5 **System.ObsoleteAttribute** . The text string that describes alternative
6 workarounds. The Boolean value that indicates whether the obsolete element
7 usage is considered an error.

8 **IsError**

9 **ToString**

10
11 [C#] public bool IsError {get;}

12 [C++] public: __property bool get_IsError();

13 [VB] Public ReadOnly Property IsError As Boolean

14 [JScript] public function get IsError() : Boolean;

15
16 *Description*

17 Gets a Boolean value indicating whether the compiler will treat usage of the
18 obsolete program element as an error.

19 **Message**

20 **ToString**

21
22 [C#] public string Message {get;}

23 [C++] public: __property String* get_Message();

24 [VB] Public ReadOnly Property Message As String

25 [JScript] public function get Message() : String;

1
2 *Description*

3 Gets the workaround message, including a description of the alternative
4 program elements.

5 TypeId

6 OperatingSystem class (System)

7 ToString

8
9
10 *Description*

11 Represents information about an operating system, such as the version and
12 platform identifier.

13 This class provides a method to copy an instance of **OperatingSystem** ,
14 and a method to return a string representation of operating system information.

15 OperatingSystem

16 *Example Syntax:*

17 ToString

18
19 [C#] public OperatingSystem(PlatformID platform, Version version);

20 [C++] public: OperatingSystem(PlatformID platform, Version* version);

21 [VB] Public Sub New(ByVal platform As PlatformID, ByVal version As Version)

22 [JScript] public function OperatingSystem(platform : PlatformID, version :
23 Version);

24
25 *Description*

1 Initializes a new instance of the **OperatingSystem** class, using the
2 specified platform identifier value and version object. A **System.PlatformID**
3 enumerated constant that indicates the operating system platform. A
4 **System.Version** object that indicates the version of the operating system.

5 Platform

6 ToString

7
8 [C#] public PlatformID Platform {get;}

9 [C++] public: __property PlatformID get_Platform();

10 [VB] Public ReadOnly Property Platform As PlatformID

11 [JScript] public function get Platform() : PlatformID;

12
13 *Description*

14 Gets a **PlatformID** value that identifies this operating system platform.

15 Version

16 ToString

17
18 [C#] public Version Version {get;}

19 [C++] public: __property Version* get_Version();

20 [VB] Public ReadOnly Property Version As Version

21 [JScript] public function get Version() : Version;

22
23 *Description*

24 Gets a **Version** object that identifies this operating system.

25 Clone

1
2 [C#] public object Clone();

3 [C++] public: __sealed Object* Clone();

4 [VB] NotOverridable Public Function Clone() As Object

5 [JScript] public function Clone() : Object;

6
7 *Description*

8 Returns an **OperatingSystem** object that is identical to this instance.

9 *Return Value:* An **OperatingSystem** object that is a copy of this instance.

10 ToString

11
12 [C#] public override string ToString();

13 [C++] public: String* ToString();

14 [VB] Overrides Public Function ToString() As String

15 [JScript] public override function ToString() : String;

16
17 *Description*

18 Converts the value of this instance to its equivalent **String** representation.

19 *Return Value:* The format of the return value is: platform

20 majorVersion.minorVersion.build.revision For example, if the operating sysem is

21 Windows 2000, the return value is: "Microsoft Windows NT 5.0.0.2195".

22 OutOfMemoryException class (System)

23 ToString

1
2
3 *Description*

4 The exception that is thrown when there is not enough memory to continue
5 the execution of a program.

6 **System.OutOfMemoryException** uses the HRESULT
7 COR_E_OUTOFMEMORY, which has the value 0x8007000E.

8 OutOfMemoryException

9 *Example Syntax:*

10 ToString

11
12 [C#] public OutOfMemoryException();

13 [C++] public: OutOfMemoryException();

14 [VB] Public Sub New()

15 [JScript] public function OutOfMemoryException(); Initializes a new instance of
16 the **System.OutOfMemoryException** class.

17
18 *Description*

19 Initializes a new instance of the **System.OutOfMemoryException** class
20 with default properties.

21 The following table shows the initial property values for an instance of
22 **System.OutOfMemoryException** .

23 OutOfMemoryException

24 *Example Syntax:*

25 ToString

```

1
2 [C#] public OutOfMemoryException(string message);
3 [C++] public: OutOfMemoryException(String* message);
4 [VB] Public Sub New(ByVal message As String)
5 [JScript] public function OutOfMemoryException(message : String);
6

```

Description

Initializes a new instance of the **System.OutOfMemoryException** class with a specified error message.

The following table shows the initial property values for an instance of **System.OutOfMemoryException**. The error message that explains the reason for the exception.

OutOfMemoryException

Example Syntax:

ToString

```

17 [C#] protected OutOfMemoryException(SerializationInfo info, StreamingContext
18 context);
19 [C++] protected: OutOfMemoryException(SerializationInfo* info,
20 StreamingContext context);
21 [VB] Protected Sub New(ByVal info As SerializationInfo, ByVal context As
22 StreamingContext)
23 [JScript] protected function OutOfMemoryException(info : SerializationInfo,
24 context : StreamingContext);
25

```

Description

Initializes a new instance of the **System.OutOfMemoryException** class with serialized data.

This constructor is called during deserialization to reconstitute the exception object transmitted over a stream. For more information, see . The object that holds the serialized object data. The contextual information about the source or destination.

OutOfMemoryException

Example Syntax:

ToString

[C#] public OutOfMemoryException(string message, Exception innerException);

[C++] public: OutOfMemoryException(String* message, Exception* innerException);

[VB] Public Sub New(ByVal message As String, ByVal innerException As Exception)

[JScript] public function OutOfMemoryException(message : String, innerException : Exception);

Description

Initializes a new instance of the **System.OutOfMemoryException** class with a specified error message and a reference to the inner exception that is the root cause of this exception.

When an **Exception** *X* is thrown as a direct result of a previous exception *Y*, the **System.Exception.InnerException** property of *X* should contain a reference to *Y*. The **InnerException** property returns the same value as was passed into the constructor, or **null** if the inner exception value was not supplied to the constructor. The error message that explains the reason for the exception. An instance of **System.Exception** that is the cause of the current **Exception**. If *innerException* is non-null, then the current **Exception** is raised in a catch block handling *innerException*.

HelpLink

HResult

InnerException

Message

Source

StackTrace

TargetSite

OverflowException class (System)

ToString

Description

The exception that is thrown when an arithmetic, casting, or conversion operation in a checked context results in an overflow.

For a result from an integral or decimal-type arithmetic operation or conversion that is outside the range of the destination type: In a checked context, a

compile-time error occurs if the operation is a constant expression. Otherwise, an

System.OverflowException is thrown if the operation is performed at run-time.

OverflowException

Example Syntax:

ToString

[C#] public OverflowException();

[C++] public: OverflowException();

[VB] Public Sub New()

[JScript] public function OverflowException(); Initializes a new instance of the

System.OverflowException class.

Description

Initializes a new instance of the **System.OverflowException** class with default properties.

The following table shows the initial property values for an instance of **System.OverflowException**.

OverflowException

Example Syntax:

ToString

[C#] public OverflowException(string message);

[C++] public: OverflowException(String* message);

[VB] Public Sub New(ByVal message As String)

[JScript] public function OverflowException(message : String);

Description

Initializes a new instance of the **System.OverflowException** class with a specified error message.

The following table shows the initial property values for an instance of **System.OverflowException**. The error message that explains the reason for the exception.

OverflowException

Example Syntax:

ToString

[C#] protected OverflowException(SerializationInfo info, StreamingContext context);

[C++] protected: OverflowException(SerializationInfo* info, StreamingContext context);

[VB] Protected Sub New(ByVal info As SerializationInfo, ByVal context As StreamingContext)

[JScript] protected function OverflowException(info : SerializationInfo, context : StreamingContext);

Description

Initializes a new instance of the **System.OverflowException** class with serialized data.

This constructor is called during deserialization to reconstitute the exception object transmitted over a stream. For more information, see . The object

that holds the serialized object data. The contextual information about the source or destination.

`OverflowException`

Example Syntax:

`ToString`

[C#] `public OverflowException(string message, Exception innerException);`

[C++] `public: OverflowException(String* message, Exception* innerException);`

[VB] `Public Sub New(ByVal message As String, ByVal innerException As Exception)`

[JScript] `public function OverflowException(message : String, innerException : Exception);`

Description

Initializes a new instance of the **System.OverflowException** class with a specified error message and a reference to the inner exception that is the root cause of this exception.

When an **Exception** *X* is thrown as a direct result of a previous exception *Y*, the **System.Exception.InnerException** property of *X* should contain a reference to *Y*. The **InnerException** property returns the same value as was passed into the constructor, or **null** if the inner exception value was not supplied to the constructor. The error message that explains the reason for the exception. An instance of **System.Exception** that is the cause of the current **Exception**. If *innerException* is non-null, then the current **Exception** is raised in a catch block handling *innerException*.

1 HelpLink
2 HResult
3 InnerException
4 Message
5 Source
6 StackTrace
7 TargetSite
8 ParamArrayAttribute class (System)
9 ToString

10
11
12 *Description*

13 Indicates that the method will allow a variable number of arguments in its
14 invocation. This class cannot be inherited.

15 A parameter array allows the specification of an unknown number of
16 arguments. A parameter array must be the last parameter in a formal parameter
17 list, and it must be a single-dimension array. A parameter array permits arguments
18 to a method to be specified in two ways: A single expression of a type that is
19 implicitly convertible to the parameter array type. The parameter array functions
20 as a value parameter.

21 ParamArrayAttribute

22 *Example Syntax:*

23 ToString

24
25 [C#] public ParamArrayAttribute();

1 [C++] public: ParamArrayAttribute();

2 [VB] Public Sub New()

3 [JScript] public function ParamArrayAttribute();

4
5 *Description*

6 Initializes a new instance of the **System.ParamArrayAttribute** class with
7 default properties.

8 TypeId

9 PlatformID enumeration (System)

10 ToString

11
12
13 *Description*

14 Describes the platforms supported by an assembly.

15 These flags are used to bind to an assembly.

16 ToString

17
18 [C#] public const PlatformID Win32NT;

19 [C++] public: const PlatformID Win32NT;

20 [VB] Public Const Win32NT As PlatformID

21 [JScript] public var Win32NT : PlatformID;

22
23 *Description*

24 The operating system is Windows NT or later.

25 ToString

[C#] public const PlatformID Win32S;

[C++] public: const PlatformID Win32S;

[VB] Public Const Win32S As PlatformID

[JScript] public var Win32S : PlatformID;

Description

The operating system is Win32s. Win32s is a layer that runs on 16-bit versions of Windows to provide access to 32-bit applications.

ToString

[C#] public const PlatformID Win32Windows;

[C++] public: const PlatformID Win32Windows;

[VB] Public Const Win32Windows As PlatformID

[JScript] public var Win32Windows : PlatformID;

Description

The operating system is Windows 95 or later.

PlatformNotSupportedException class (System)

ToString

Description

The exception that is thrown when a feature does not run on a particular platform.

PlatformNotSupportedException uses the HRESULT
COR_E_PLATFORMNOTSUPPORTED, which has the value 0x80131539.

PlatformNotSupportedException

Example Syntax:

ToString

[C#] public **PlatformNotSupportedException**();

[C++] public: **PlatformNotSupportedException**();

[VB] Public Sub New()

[JScript] public function **PlatformNotSupportedException**(); Initializes a new
instance of the **System.PlatformNotSupportedException** class.

Description

Initializes a new instance of the **System.PlatformNotSupportedException**
class with default properties.

When an instance of the **System.PlatformNotSupportedException** class
is created by a call to this constructor, the following properties are initialized to
the specified values: Property Value **System.Exception.InnerException** null .

PlatformNotSupportedException

Example Syntax:

ToString

[C#] public **PlatformNotSupportedException**(string message);

[C++] public: **PlatformNotSupportedException**(String* message);

[VB] Public Sub New(ByVal message As String)

1 [JScript] public function PlatformNotSupportedException(message : String);

3 *Description*

4 Initializes a new instance of the **System.PlatformNotSupportedException**
5 class with a specified error message.

6 When an instance of the **System.PlatformNotSupportedException** class
7 is created by a call to this constructor, the following properties are initialized to
8 the specified values: Property Value **System.Exception.InnerException** null . The
9 text message that explains the reason for the exception.

10 PlatformNotSupportedException

11 *Example Syntax:*

12 ToString

14 [C#] protected PlatformNotSupportedException(SerializationInfo info,
15 StreamingContext context);

16 [C++] protected: PlatformNotSupportedException(SerializationInfo* info,
17 StreamingContext context);

18 [VB] Protected Sub New(ByVal info As SerializationInfo, ByVal context As
19 StreamingContext)

20 [JScript] protected function PlatformNotSupportedException(info :
21 SerializationInfo, context : StreamingContext);

23 *Description*

24 Initializes a new instance of the **System.PlatformNotSupportedException**
25 class with serialized data. The **System.Runtime.Serialization.SerializationInfo**

that holds the serialized object data about the exception being thrown. The **System.Runtime.Serialization.StreamingContext** that contains contextual information about the source or destination.

PlatformNotSupportedException

Example Syntax:

ToString

[C#] public PlatformNotSupportedException(string message, Exception inner);

[C++] public: PlatformNotSupportedException(String* message, Exception* inner);

[VB] Public Sub New(ByVal message As String, ByVal inner As Exception)

[JScript] public function PlatformNotSupportedException(message : String, inner : Exception);

Description

Initializes a new instance of the **System.PlatformNotSupportedException** class with a specified error message and a reference to the inner exception that is the root cause of this exception.

When an **Exception** *X* is thrown as a direct result of a previous exception *Y*, the **System.Exception.InnerException** property of *X* should contain a reference to *Y*. The **InnerException** property returns the same value as was passed into the constructor, or **null** if the inner exception value was not supplied to the constructor. The text message that explains the reason for the exception. An instance of **System.Exception** that is the cause of the current **Exception**. If the

1 *inner* parameter is non-null, then the current **Exception** is raised in a catch block
2 handling *inner* .

3 HelpLink

4 HResult

5 InnerException

6 Message

7 Source

8 StackTrace

9 TargetSite

10 Random class (System)

11 ToString

12
13
14 *Description*

15 Represents a pseudo-random number generator, a device that produces a
16 sequence of numbers that meet certain statistical requirements for randomness.

17 Pseudo-random numbers are chosen with equal probability from a finite set
18 of numbers. The chosen numbers are not completely random because a definite
19 mathematical algorithm is used to select them, but they are sufficiently random for
20 practical purposes.

21 Random

22 *Example Syntax:*

23 ToString

24
25 [C#] public Random();

1 [C++] public: Random();

2 [VB] Public Sub New()

3 [JScript] public function Random(); Initializes a new instance of the **Random**
4 class.

5
6 *Description*

7 Initializes a new instance of the **Random** class, using a time-dependent
8 default seed value.

9 The distribution of the generated numbers is uniform; each number is
10 equally likely to be returned.

11 Random

12 *Example Syntax:*

13 ToString

14
15 [C#] public Random(int Seed);

16 [C++] public: Random(int Seed);

17 [VB] Public Sub New(ByVal Seed As Integer)

18 [JScript] public function Random(Seed : int);

19
20 *Description*

21 Initializes a new instance of the **Random** class, using the specified seed
22 value.

23 If your application requires different random number sequences, invoke this
24 constructor repeatedly with different seed values. One way to produce a unique
25 seed value is to make it time-dependent. For example, derive the seed value from

the system clock. A number used to calculate a starting value for the pseudo-random number sequence.

Next

[C#] public virtual int Next();

[C++] public: virtual int Next();

[VB] Overridable Public Function Next() As Integer

[JScript] public function Next() : int; Returns a random number.

Description

Returns a positive random number.

Return Value: A number greater than or equal to zero and less than

System.Int32.MaxValue .

Next

[C#] public virtual int Next(int maxValue);

[C++] public: virtual int Next(int maxValue);

[VB] Overridable Public Function Next(ByVal maxValue As Integer) As Integer

[JScript] public function Next(maxValue : int) : int;

Description

Returns a positive random number less than the specified maximum.

Return Value: A number greater than or equal to zero, and less than *maxValue* .

The upper bound of the random number to be generated.

Next

1
2 [C#] public virtual int Next(int minValue, int maxValue);

3 [C++] public: virtual int Next(int minValue, int maxValue);

4 [VB] Overridable Public Function Next(ByVal minValue As Integer, ByVal
5 maxValue As Integer) As Integer

6 [JScript] public function Next(minValue : int, maxValue : int) : int;

7
8 *Description*

9 Returns a random number within a specified range.

10 *Return Value:* A number greater than or equal to *minValue* and less than *maxValue*
11 . If *minValue* equals *maxValue* , *minValue* is returned. The lower bound of the
12 random number returned. The upper bound of the random number returned.

13 NextBytes

14
15 [C#] public virtual void NextBytes(byte[] buffer);

16 [C++] public: virtual void NextBytes(unsigned char buffer __gc[]);

17 [VB] Overridable Public Sub NextBytes(ByVal buffer() As Byte)

18 [JScript] public function NextBytes(buffer : Byte[]);

19
20 *Description*

21 Fills the elements of a specified array of bytes with random numbers.

22 Each element of the array of bytes is set to a random number greater than or
23 equal to zero, and less than or equal to **System.Byte.MaxValue** . An array of
24 bytes to contain random numbers.

25 NextDouble

1
2 [C#] public virtual double NextDouble();

3 [C++] public: virtual double NextDouble();

4 [VB] Overridable Public Function NextDouble() As Double

5 [JScript] public function NextDouble() : double;

6
7 *Description*

8 Returns a random number between 0.0 and 1.0.

9 *Return Value:* A double-precision floating point number greater than or equal to
10 0.0, and less than 1.0.

11 This method is the **public** version of the **protected** method,

12 **System.Random.Sample** .

13 Sample

14
15 [C#] protected virtual double Sample();

16 [C++] protected: virtual double Sample();

17 [VB] Overridable Protected Function Sample() As Double

18 [JScript] protected function Sample() : double;

19
20 *Description*

21 Returns a random number between 0.0 and 1.0.

22 *Return Value:* A double-precision floating point number greater than or equal to
23 0.0, and less than 1.0.

24 Create a derived class of **Random** to override this method and produce a
25 different distribution.

RankException class (System)

ToString

Description

The exception that is thrown when an array with the wrong number of dimensions is passed to a method.

System.RankException uses the HRESULT COR_E_RANK, which has the value 0x80131517.

RankException

Example Syntax:

ToString

[C#] public RankException();

[C++] public: RankException();

[VB] Public Sub New()

[JScript] public function RankException(); Initializes a new instance of the **System.RankException** class.

Description

Initializes a new instance of the **System.RankException** class with default properties.

The following table shows the initial property values for an instance of **System.RankException**.

RankException

Example Syntax:

ToString

[C#] public RankException(string message);

[C++] public: RankException(String* message);

[VB] Public Sub New(ByVal message As String)

[JScript] public function RankException(message : String);

Description

Initializes a new instance of the **System.RankException** class with a specified error message.

The following table shows the initial property values for an instance of **System.RankException**. The error message that explains the reason for the exception.

RankException

Example Syntax:

ToString

[C#] protected RankException(SerializationInfo info, StreamingContext context);

[C++] protected: RankException(SerializationInfo* info, StreamingContext context);

[VB] Protected Sub New(ByVal info As SerializationInfo, ByVal context As StreamingContext)

[JScript] protected function RankException(info : SerializationInfo, context : StreamingContext);

Description

Initializes a new instance of the **System.RankException** class with serialized data.

This constructor is called during deserialization to reconstitute the exception object transmitted over a stream. For more information, see . The object that holds the serialized object data. The contextual information about the source or destination.

RankException

Example Syntax:

ToString

[C#] public RankException(string message, Exception innerException);

[C++] public: RankException(String* message, Exception* innerException);

[VB] Public Sub New(ByVal message As String, ByVal innerException As Exception)

[JScript] public function RankException(message : String, innerException : Exception);

Description

Initializes a new instance of the **System.RankException** class with a specified error message and a reference to the inner exception that is the root cause of this exception.

When an **Exception** *X* is thrown as a direct result of a previous exception *Y*, the **System.Exception.InnerException** property of *X* should contain a reference

1 to *Y* . The **InnerException** property returns the same value as was passed into the
2 constructor, or **null** if the inner exception value was not supplied to the
3 constructor. The error message that explains the reason for the exception. An
4 instance of **System.Exception** that is the cause of the current **Exception**. If
5 *innerException* is non-null, then the current **Exception** is raised in a catch block
6 handling *innerException* .

7 HelpLink

8 HResult

9 InnerException

10 Message

11 Source

12 StackTrace

13 TargetSite

14 ResolveEventArgs class (System)

15 ToString

17
18 *Description*

19 Provides data for the **System.AppDomain.TypeResolve** ,
20 **System.AppDomain.ResourceResolve** , and
21 **System.AppDomain.AssemblyResolve** events.

22 ResolveEventArgs

23 *Example Syntax:*

24 ToString


```

1
2 [C#] public ResolveEventArgs(string name);
3 [C++] public: ResolveEventArgs(String* name);
4 [VB] Public Sub New(ByVal name As String)
5 [JScript] public function ResolveEventArgs(name : String);
6

```

Description

Initializes a new instance of the **ResolveEventArgs** class.

This constructor is typically only called by the common language runtime.

The name of an item to resolve.

Name

ToString

```

13
14 [C#] public string Name {get;}
15 [C++] public: __property String* get_Name();
16 [VB] Public ReadOnly Property Name As String
17 [JScript] public function get Name() : String;
18

```

Description

The name of the item to be resolved.

ResolveEventHandler delegate (System)

ToString

Description

Represents the method that will handle the **System.AppDomain.TypeResolve** , **System.AppDomain.ResourceResolve** , and **System.AppDomain.AssemblyResolve** events of an **System.AppDomain** . The source of the event. A **System.ResolveEventArgs** that contains the event data.

If the runtime class loader cannot resolve a reference to an assembly, type or a resource through normal means, the corresponding events are raised to give the callback a chance to tell the runtime which assembly the referenced assembly, type or resource is in.

RuntimeArgumentHandle structure (System)

ToString

Description

References a variable-length argument list.

This class has no members, and exists solely to support C/C++ programming language functions that take a variable number of parameters.

RuntimeFieldHandle structure (System)

ToString

Description

The RuntimeFieldHandle is a handle to the internal metadata representation of a field.

Value

ToString

1
2 [C#] public IntPtr Value {get;}

3 [C++] public: __property IntPtr get_Value();

4 [VB] Public ReadOnly Property Value As IntPtr

5 [JScript] public function get Value() : IntPtr;

6
7 *Description*

8 The value of the handle.

9 GetObjectData

10
11 [C#] public void GetObjectData(SerializationInfo info, StreamingContext
12 context);

13 [C++] public: __sealed void GetObjectData(SerializationInfo* info,
14 StreamingContext context);

15 [VB] NotOverridable Public Sub GetObjectData(ByVal info As SerializationInfo,
16 ByVal context As StreamingContext)

17 [JScript] public function GetObjectData(info : SerializationInfo, context :
18 StreamingContext);

19
20 *Description*

21 Returns a SerializationInfo completely populated with all the data needed to
22 reinstantiate the object at the other end of serialization. The object to be populated
23 with serialization information. The destination context of the serialization.

24 RuntimeMethodHandle structure (System)

25 ToString

Description

The RuntimeMethodHandle is a handle to the internal metadata representation of a method.

Value

ToString

[C#] public IntPtr Value {get;}

[C++] public: __property IntPtr get_Value();

[VB] Public ReadOnly Property Value As IntPtr

[JScript] public function get Value() : IntPtr;

Description

The value of the handle.

GetFunctionPointer

[C#] public IntPtr GetFunctionPointer();

[C++] public: IntPtr GetFunctionPointer();

[VB] Public Function GetFunctionPointer() As IntPtr

[JScript] public function GetFunctionPointer() : IntPtr;

GetObjectData

[C#] public void GetObjectData(SerializationInfo info, StreamingContext context);

```

1 [C++] public: __sealed void GetObjectData(SerializationInfo* info,
2 StreamingContext context);
3 [VB] NotOverridable Public Sub GetObjectData(ByVal info As SerializationInfo,
4 ByVal context As StreamingContext)
5 [JScript] public function GetObjectData(info : SerializationInfo, context :
6 StreamingContext);
7

```

Description

Returns a `SerializationInfo` completely populated with all the data needed to reinstantiate the object at the other end of serialization. The object to be populated with serialization information. The destination context of the serialization.

`RuntimeTypeHandle` structure (System)

`ToString`

Description

The `RuntimeTypeHandle` is a handle to the internal metadata representation of a type.

`Value`

`ToString`

```

22 [C#] public IntPtr Value {get;}
23 [C++] public: __property IntPtr get_Value();
24 [VB] Public ReadOnly Property Value As IntPtr
25 [JScript] public function get Value() : IntPtr;

```

1
2 *Description*

3 The value of the handle.

4 GetObjectData

5
6 [C#] public void GetObjectData(SerializationInfo info, StreamingContext
7 context);

8 [C++] public: __sealed void GetObjectData(SerializationInfo* info,
9 StreamingContext context);

10 [VB] NotOverridable Public Sub GetObjectData(ByVal info As SerializationInfo,
11 ByVal context As StreamingContext)

12 [JScript] public function GetObjectData(info : SerializationInfo, context :
13 StreamingContext);

14
15 *Description*

16 Returns a SerializationInfo completely populated with all the data needed to
17 reinstantiate the object at the other end of serialization. The object to be populated
18 with serialization information. The destination context of the serialization.

19 SByte structure (System)

20 ToString

21
22
23 *Description*

24 Represents an 8-bit signed integer.

The **SByte** value type represents integers with values ranging from negative 128 to positive 127.

ToString

[C#] public const sbyte MaxValue;

[C++] public: const char MaxValue;

[VB] Public Const MaxValue As SByte

[JScript] public var MaxValue : SByte;

Description

A constant representing the largest possible value of **SByte** .

The value of this constant is 127; that is, hexadecimal 0x7F.

ToString

[C#] public const sbyte MinValue;

[C++] public: const char MinValue;

[VB] Public Const MinValue As SByte

[JScript] public var MinValue : SByte;

Description

A constant representing the smallest possible value of **SByte** .

The value of this constant is -128; that is, hexadecimal 0x80.

CompareTo

[C#] public int CompareTo(object obj);

1 [C++] public: __sealed int CompareTo(Object* obj);

2 [VB] NotOverridable Public Function CompareTo(ByVal obj As Object) As

3 Integer

4 [JScript] public function CompareTo(obj : Object) : int;

6 *Description*

7 Compares this instance to a specified object and returns an indication of
8 their relative values.

9 *Return Value:* A signed number indicating the relative values of this instance and
10 *obj* .

11 Any instance of **SByte** , regardless of its value, is considered greater than
12 **null** . An object to compare, or **null**.

13 Equals

15 [C#] public override bool Equals(object obj);

16 [C++] public: bool Equals(Object* obj);

17 [VB] Overrides Public Function Equals(ByVal obj As Object) As Boolean

18 [JScript] public override function Equals(obj : Object) : Boolean;

20 *Description*

21 Returns a value indicating whether this instance is equal to a specified
22 object.

23 *Return Value:* **true** if *obj* is an instance of **SByte** and equals the value of this
24 instance; otherwise, **false** . An object to compare with this instance.

25 GetHashCode

1
2 [C#] public override int GetHashCode();

3 [C++] public: int GetHashCode();

4 [VB] Overrides Public Function GetHashCode() As Integer

5 [JScript] public override function GetHashCode() : int;

6
7 *Description*

8 Returns the hash code for this instance.

9 *Return Value:* A 32-bit signed integer hash code.

10 **GetTypeCode**

11
12 [C#] public TypeCode GetTypeCode();

13 [C++] public: __sealed TypeCode GetTypeCode();

14 [VB] NotOverridable Public Function GetTypeCode() As TypeCode

15 [JScript] public function GetTypeCode() : TypeCode;

16
17 *Description*

18 Returns the **TypeCode** for value type **SByte** .

19 *Return Value:* The enumerated constant, **System.TypeCode.SByte** .

20 **Parse**

21
22 [C#] public static sbyte Parse(string s);

23 [C++] public: static char Parse(String* s);

24 [VB] Public Shared Function Parse(ByVal s As String) As SByte

25 [JScript] public static function Parse(s : String) : SByte; Converts the **String**

1 representation of a number to its 8-bit signed integer equivalent.

3 *Description*

4 Converts the **String** representation of a number to its 8-bit signed integer
5 equivalent.

6 *Return Value:* An 8-bit signed integer equivalent to the number contained in *s* .

7 *s* contains a number of the form: [ws][sign]digits[ws] Items in square
8 brackets ('[' and ']') are optional, and other items are as follows. A **System.String**
9 containing a number to convert.

10 *Parse*

11
12 [C#] public static sbyte Parse(string s, IFormatProvider provider);

13 [C++] public: static char Parse(String* s, IFormatProvider* provider);

14 [VB] Public Shared Function Parse(ByVal s As String, ByVal provider As
15 IFormatProvider) As SByte

16 [JScript] public static function Parse(s : String, provider : IFormatProvider) :
17 SByte;

18 19 *Description*

20 Converts the **String** representation of a number in a specified culture-
21 specific format to its 8-bit signed integer equivalent.

22 *Return Value:* An 8-bit signed integer equivalent to the number specified in *s* .

23 *s* contains a number of the form: [ws][sign]digits[ws] Items in square
24 brackets ('[' and ']') are optional, and other items are as follows. A **System.String**
25

1 containing a number to convert. An **System.IFormatProvider** interface
2 implementation which supplies culture-specific formatting information about *s*.

3 Parse

4
5 [C#] public static sbyte Parse(string s, NumberStyles style);

6 [C++] public: static char Parse(String* s, NumberStyles style);

7 [VB] Public Shared Function Parse(ByVal s As String, ByVal style As
8 NumberStyles) As SByte

9 [JScript] public static function Parse(s : String, style : NumberStyles) : SByte;

10 11 *Description*

12 Converts the **String** representation of a number in a specified style to its 8-
13 bit signed integer equivalent.

14 *Return Value:* An 8-bit signed integer equivalent to the number specified in *s* .

15 *s* contains a number of the form: [ws][sign]digits[ws] Items in square
16 brackets ('[' and ']') are optional, and other items are as follows. A **System.String**
17 containing a number to convert. The combination of one or more
18 **System.Globalization.NumberStyles** constants that indicate the permitted format
19 of *s*.

20 Parse

21
22 [C#] public static sbyte Parse(string s, NumberStyles style, IFormatProvider
23 provider);

24 [C++] public: static char Parse(String* s, NumberStyles style, IFormatProvider*
25 provider);

```

1 [VB] Public Shared Function Parse(ByVal s As String, ByVal style As
2 NumberStyles, ByVal provider As IFormatProvider) As SByte
3 [JScript] public static function Parse(s : String, style : NumberStyles, provider :
4 IFormatProvider) : SByte;

```

Description

Converts the **String** representation of a number in a specified style and culture-specific format to its 8-bit signed integer equivalent.

Return Value: An 8-bit signed integer equivalent to the number specified in *s*.

s contains a number of the form: [ws][sign]digits[ws] Items in square brackets ('[' and ']') are optional, and other items are as follows. A **System.String** containing a number to convert. The combination of one or more **System.Globalization.NumberStyles** constants that indicate the permitted format of *s*. An **System.IFormatProvider** interface implementation which supplies culture-specific formatting information about *s*.

ICconvertible.ToBoolean

```

18 [C#] bool IConvertible.ToBoolean(IFormatProvider provider);

```

```

19 [C++] bool IConvertible::ToBoolean(IFormatProvider* provider);

```

```

20 [VB] Function ToBoolean(ByVal provider As IFormatProvider) As Boolean

```

Implements **ICconvertible.ToBoolean**

```

22 [JScript] function IConvertible.ToBoolean(provider : IFormatProvider) : Boolean;

```

ICconvertible.ToByte

```

25 [C#] byte IConvertible.ToByte(IFormatProvider provider);

```

```

1  [C++] unsigned char IConvertible::ToByte(IFormatProvider* provider);
2  [VB] Function ToByte(ByVal provider As IFormatProvider) As Byte Implements
3  IConvertible.ToByte
4  [JScript] function IConvertible.ToByte(provider : IFormatProvider) : Byte;
5      IConvertible.ToChar
6
7  [C#] char IConvertible.ToChar(IFormatProvider provider);
8  [C++] __wchar_t IConvertible::ToChar(IFormatProvider* provider);
9  [VB] Function ToChar(ByVal provider As IFormatProvider) As Char Implements
10 IConvertible.ToChar
11 [JScript] function IConvertible.ToChar(provider : IFormatProvider) : Char;
12     IConvertible.ToDateTime
13
14 [C#] DateTime IConvertible.ToDateTime(IFormatProvider provider);
15 [C++] DateTime IConvertible::ToDateTime(IFormatProvider* provider);
16 [VB] Function ToDateTime(ByVal provider As IFormatProvider) As DateTime
17 Implements IConvertible.ToDateTime
18 [JScript] function IConvertible.ToDateTime(provider : IFormatProvider) :
19 DateTime;
20     IConvertible.ToDecimal
21
22 [C#] decimal IConvertible.ToDecimal(IFormatProvider provider);
23 [C++] Decimal IConvertible::ToDecimal(IFormatProvider* provider);
24 [VB] Function ToDecimal(ByVal provider As IFormatProvider) As Decimal
25

```

1 Implements IConvertible.ToDecimal

2 [JScript] function IConvertible.ToDecimal(provider : IFormatProvider) : Decimal;

3 IConvertible.ToDouble

4
5 [C#] double IConvertible.ToDouble(IFormatProvider provider);

6 [C++] double IConvertible::ToDouble(IFormatProvider* provider);

7 [VB] Function ToDouble(ByVal provider As IFormatProvider) As Double

8 Implements IConvertible.ToDouble

9 [JScript] function IConvertible.ToDouble(provider : IFormatProvider) : double;

10 IConvertible.ToInt16

11
12 [C#] short IConvertible.ToInt16(IFormatProvider provider);

13 [C++] short IConvertible::ToInt16(IFormatProvider* provider);

14 [VB] Function ToInt16(ByVal provider As IFormatProvider) As Short

15 Implements IConvertible.ToInt16

16 [JScript] function IConvertible.ToInt16(provider : IFormatProvider) : Int16;

17 IConvertible.ToInt32

18
19 [C#] int IConvertible.ToInt32(IFormatProvider provider);

20 [C++] int IConvertible::ToInt32(IFormatProvider* provider);

21 [VB] Function ToInt32(ByVal provider As IFormatProvider) As Integer

22 Implements IConvertible.ToInt32

23 [JScript] function IConvertible.ToInt32(provider : IFormatProvider) : int;

24 IConvertible.ToInt64

```

1
2 [C#] long IConvertible.ToInt64(IFormatProvider provider);
3 [C++] __int64 IConvertible::ToInt64(IFormatProvider* provider);
4 [VB] Function ToInt64(ByVal provider As IFormatProvider) As Long Implements
5 IConvertible.ToInt64
6 [JScript] function IConvertible.ToInt64(provider : IFormatProvider) : long;
7     IConvertible.ToSByte
8
9 [C#] sbyte IConvertible.ToSByte(IFormatProvider provider);
10 [C++] char IConvertible::ToSByte(IFormatProvider* provider);
11 [VB] Function ToSByte(ByVal provider As IFormatProvider) As SByte
12 Implements IConvertible.ToSByte
13 [JScript] function IConvertible.ToSByte(provider : IFormatProvider) : SByte;
14     IConvertible.ToSingle
15
16 [C#] float IConvertible.ToSingle(IFormatProvider provider);
17 [C++] float IConvertible::ToSingle(IFormatProvider* provider);
18 [VB] Function ToSingle(ByVal provider As IFormatProvider) As Single
19 Implements IConvertible.ToSingle
20 [JScript] function IConvertible.ToSingle(provider : IFormatProvider) : float;
21     IConvertible.ToType
22
23 [C#] object IConvertible.ToType(Type type, IFormatProvider provider);
24 [C++] Object* IConvertible::ToType(Type* type, IFormatProvider* provider);
25 [VB] Function ToType(ByVal type As Type, ByVal provider As IFormatProvider)

```

```

1 As Object Implements IConvertible.ToType
2 [JScript] function IConvertible.ToType(type : Type, provider : IFormatProvider) :
3 Object;
4     IConvertible.ToUInt16
5
6 [C#] ushort IConvertible.ToUInt16(IFormatProvider provider);
7 [C++] unsigned short IConvertible::ToUInt16(IFormatProvider* provider);
8 [VB] Function ToUInt16(ByVal provider As IFormatProvider) As UInt16
9 Implements IConvertible.ToUInt16
10 [JScript] function IConvertible.ToUInt16(provider : IFormatProvider) : UInt16;
11     IConvertible.ToUInt32
12
13 [C#] uint IConvertible.ToUInt32(IFormatProvider provider);
14 [C++] unsigned int IConvertible::ToUInt32(IFormatProvider* provider);
15 [VB] Function ToUInt32(ByVal provider As IFormatProvider) As UInt32
16 Implements IConvertible.ToUInt32
17 [JScript] function IConvertible.ToUInt32(provider : IFormatProvider) : UInt32;
18     IConvertible.ToUInt64
19
20 [C#] ulong IConvertible.ToUInt64(IFormatProvider provider);
21 [C++] unsigned __int64 IConvertible::ToUInt64(IFormatProvider* provider);
22 [VB] Function ToUInt64(ByVal provider As IFormatProvider) As UInt64
23 Implements IConvertible.ToUInt64
24 [JScript] function IConvertible.ToUInt64(provider : IFormatProvider) : UInt64;
25     ToString

```


1
2 [C#] public override string ToString();

3 [C++] public: String* ToString();

4 [VB] Overrides Public Function ToString() As String

5 [JScript] public override function ToString() : String; Converts the numeric value
6 of this instance to its equivalent **String** representation.

7
8 *Description*

9 Converts the numeric value of this instance to its equivalent **String**
10 representation.

11 *Return Value:* The **System.String** representation of the value of this instance,
12 consisting of a negative sign if the value is negative, and a sequence of digits
13 ranging from 0 to 9 with no leading zeroes.

14 The return value is formatted with the general format specifier ("G") and
15 the **System.Globalization.NumberFormatInfo** for the current culture.

16 **ToString**

17
18 [C#] public string ToString(IFormatProvider provider);

19 [C++] public: __sealed String* ToString(IFormatProvider* provider);

20 [VB] NotOverridable Public Function ToString(ByVal provider As
21 IFormatProvider) As String

22 [JScript] public function ToString(provider : IFormatProvider) : String;

23
24 *Description*

Converts the numeric value of this instance to its equivalent **String** representation using the specified culture-specific format information.

Return Value: The **System.String** representation of the value of this instance as specified by *provider* .

This instance is formatted with the general format specifier ("G"). An **System.IFormatProvider** interface implementation which supplies culture-specific formatting information.

ToString

[C#] public string ToString(string format);

[C++] public: String* ToString(String* format);

[VB] Public Function ToString(ByVal format As String) As String

[JScript] public function ToString(format : String) : String;

Description

Converts the numeric value of this instance to its equivalent **String** representation, using the specified format.

Return Value: The **System.String** representation of the value of this instance as specified by *format* .

If *format* is **null** or an empty string, the return value of this instance is formatted with the general format specifier ("G"). A format string.

ToString

[C#] public string ToString(string format, IFormatProvider provider);

[C++] public: __sealed String* ToString(String* format, IFormatProvider*

1 provider);

2 [VB] NotOverridable Public Function ToString(ByVal format As String, ByVal
3 provider As IFormatProvider) As String

4 [JScript] public function ToString(format : String, provider : IFormatProvider) :
5 String;

6 7 *Description*

8 Converts the numeric value of this instance to its equivalent **String**
9 representation using the specified format and culture-specific format information.

10 *Return Value:* The **System.String** representation of the value of this instance as
11 specified by *format* and *provider* .

12 If *format* is **null** or an empty string, the return value for this instance is
13 formatted with the general format specifier ("G"). A format specification. An
14 **System.IFormatProvider** interface implementation which supplies culture-
15 specific formatting information.

16 SerializableAttribute class (System)

17 ToString

18 19 20 *Description*

21 Indicates that a class can be serialized. This class cannot be inherited.

22 Apply the **System.SerializableAttribute** attribute to a class to indicate it
23 can be serialized. The common language runtime throws
24 **System.Runtime.Serialization.SerializationException** if any class in the graph
25

of objects being serialized does not have the **System.SerializableAttribute** attribute applied.

SerializableAttribute

Example Syntax:

ToString

[C#] public SerializableAttribute();

[C++] public: SerializableAttribute();

[VB] Public Sub New()

[JScript] public function SerializableAttribute();

Description

Initializes a new instance of the **System.SerializableAttribute** class.

TypeId

Single structure (System)

ToString

Description

Represents a single-precision floating point number.

The **Single** value type represents a single-precision 32-bit number with values ranging from negative 3.402823e38 to positive 3.402823e38, as well as positive or negative zero, **System.Single.PositiveInfinity** , **System.Single.NegativeInfinity** , and Not-a-Number (**System.Single.NaN**).

ToString

```

1
2 [C#] public const float Epsilon;
3 [C++] public: const float Epsilon;
4 [VB] Public Const Epsilon As Single
5 [JScript] public var Epsilon : float;
6

```

Description

A constant representing the smallest positive **Single** greater than zero.

The value of this constant is 1.4e-45.

ToString

```

11
12 [C#] public const float MaxValue;
13 [C++] public: const float MaxValue;
14 [VB] Public Const MaxValue As Single
15 [JScript] public var MaxValue : float;
16

```

Description

A constant representing the largest possible value of **Single** .

The value of this constant is positive 3.402823e38.

ToString

```

21
22 [C#] public const float MinValue;
23 [C++] public: const float MinValue;
24 [VB] Public Const MinValue As Single
25 [JScript] public var MinValue : float;

```

1
2 *Description*

3 A constant representing the smallest possible value of **Single** .

4 The value of this constant is negative 3.402823e38.

5 ToString

6
7 [C#] public const float NaN;

8 [C++] public: const float NaN;

9 [VB] Public Const NaN As Single

10 [JScript] public var NaN : float;

11
12 *Description*

13 A constant representing Not-a-Number (**NaN**).

14 The value of this constant is the result of dividing zero by zero.

15 ToString

16
17 [C#] public const float NegativeInfinity;

18 [C++] public: const float NegativeInfinity;

19 [VB] Public Const NegativeInfinity As Single

20 [JScript] public var NegativeInfinity : float;

21
22 *Description*

23 A constant representing negative infinity.

24 The value of this constant is the result of dividing a negative number by

25 zero.

ToString

[C#] public const float PositiveInfinity;
[C++] public: const float PositiveInfinity;
[VB] Public Const PositiveInfinity As Single
[JScript] public var PositiveInfinity : float;

Description

A constant representing positive infinity.

The value of this constant is the result of dividing a positive number by zero.

CompareTo

[C#] public int CompareTo(object value);
[C++] public: __sealed int CompareTo(Object* value);
[VB] NotOverridable Public Function CompareTo(ByVal value As Object) As Integer
[JScript] public function CompareTo(value : Object) : int;

Description

Compares this instance to a specified object and returns an indication of their relative values.

Return Value: A signed number indicating the relative values of this instance and *value*.

Any instance of **Single** , regardless of its value, is considered greater than **null** . An object to compare, or **null**.

Equals

[C#] public override bool Equals(object obj);

[C++] public: bool Equals(Object* obj);

[VB] Overrides Public Function Equals(ByVal obj As Object) As Boolean

[JScript] public override function Equals(obj : Object) : Boolean;

Description

Returns a value indicating whether this instance is equal to a specified object.

Return Value: **true** if *obj* is an instance of **Single** and equals the value of this instance; otherwise, **false** . An object to compare with this instance.

GetHashCode

[C#] public override int GetHashCode();

[C++] public: int GetHashCode();

[VB] Overrides Public Function GetHashCode() As Integer

[JScript] public override function GetHashCode() : int;

Description

Returns the hash code for this instance.

Return Value: A 32-bit signed integer hash code.

GetTypeCode

1
2 [C#] public TypeCode GetTypeCode();

3 [C++] public: __sealed TypeCode GetTypeCode();

4 [VB] NotOverridable Public Function GetTypeCode() As TypeCode

5 [JScript] public function GetTypeCode() : TypeCode;

6
7 *Description*

8 Returns the **TypeCode** for value type **Single** .

9 *Return Value:* The enumerated constant, **System.TypeCode.Single** .

10 **IsInfinity**

11
12 [C#] public static bool IsInfinity(float f);

13 [C++] public: static bool IsInfinity(float f);

14 [VB] Public Shared Function IsInfinity(ByVal f As Single) As Boolean

15 [JScript] public static function IsInfinity(f : float) : Boolean;

16
17 *Description*

18 Returns a value indicating whether the specified number evaluates to either
19 negative or positive infinity.

20 *Return Value:* **true** if *f* evaluates to negative or positive infinity; otherwise, **false** .

21 A single-precision floating point number.

22 **IsNaN**

23
24 [C#] public static bool IsNaN(float f);

25 [C++] public: static bool IsNaN(float f);

1 [VB] Public Shared Function IsNaN(ByVal f As Single) As Boolean

2 [JScript] public static function IsNaN(f : float) : Boolean;

3
4 *Description*

5 Returns a value indicating whether the specified number evaluates to Not-a-
6 Number (NaN).

7 *Return Value:* **true** if *f* evaluates to NaN; otherwise, **false** . A single-precision
8 floating point number.

9 IsNegativeInfinity

10
11 [C#] public static bool IsNegativeInfinity(float f);

12 [C++] public: static bool IsNegativeInfinity(float f);

13 [VB] Public Shared Function IsNegativeInfinity(ByVal f As Single) As Boolean

14 [JScript] public static function IsNegativeInfinity(f : float) : Boolean;

15
16 *Description*

17 Returns a value indicating whether the specified number evaluates to
18 negative infinity.

19 *Return Value:* **true** if *f* evaluates to negative infinity; otherwise, **false** . A single-
20 precision floating point number.

21 IsPositiveInfinity

22
23 [C#] public static bool IsPositiveInfinity(float f);

24 [C++] public: static bool IsPositiveInfinity(float f);

25 [VB] Public Shared Function IsPositiveInfinity(ByVal f As Single) As Boolean

[JScript] public static function IsPositiveInfinity(f : float) : Boolean;

Description

Returns a value indicating whether the specified number evaluates to positive infinity.

Return Value: **true** if *f* evaluates to positive infinity; otherwise, **false** . A single-precision floating point number.

Parse

[C#] public static float Parse(string s);

[C++] public: static float Parse(String* s);

[VB] Public Shared Function Parse(ByVal s As String) As Single

[JScript] public static function Parse(s : String) : float; Converts the **String** representation of a number to its single-precision floating point number equivalent.

Description

Converts the **String** representation of a number to its single-precision floating point number equivalent.

Return Value: A single-precision floating point number equivalent to the numeric value or symbol specified in *s* .

s can contain

System.Globalization.NumberFormatInfo.PositiveInfinitySymbol ,

System.Globalization.NumberFormatInfo.NegativeInfinitySymbol ,

System.Globalization.NumberFormatInfo.NaNSymbol , or a string of the form:

[ws][sign]integral-digits[.[fractional-digits]][e[sign]exponential-digits][ws]

Optional items are framed in square brackets ('[' and ']'). Items containing the term

"digits" consist of a series of numeric characters ranging from 0 to 9. A

System.String containing a number to convert.

Parse

[C#] public static float Parse(string s, IFormatProvider provider);

[C++] public: static float Parse(String* s, IFormatProvider* provider);

[VB] Public Shared Function Parse(ByVal s As String, ByVal provider As

IFormatProvider) As Single

[JScript] public static function Parse(s : String, provider : IFormatProvider) : float;

Description

Converts the **String** representation of a number in a specified culture-specific format to its single-precision floating point number equivalent.

Return Value: A single-precision floating point number equivalent to the numeric value or symbol specified in *s*.

s can contain

System.Globalization.NumberFormatInfo.PositiveInfinitySymbol,

System.Globalization.NumberFormatInfo.NegativeInfinitySymbol,

System.Globalization.NumberFormatInfo.NaNSymbol, or a string of the form:

[ws][sign]integral-digits[.[fractional-digits]][e[sign]exponential-digits][ws]

Optional items are framed in square brackets ('[' and ']'). Items containing the term

"digits" consist of a series of numeric characters ranging from 0 to 9. A

System.String containing a number to convert. An **System.IFormatProvider**

interface implementation which supplies culture-specific formatting information about *s*.

Parse

[C#] public static float Parse(string s, NumberStyles style);

[C++] public: static float Parse(String* s, NumberStyles style);

[VB] Public Shared Function Parse(ByVal s As String, ByVal style As NumberStyles) As Single

[JScript] public static function Parse(s : String, style : NumberStyles) : float;

Description

Converts the **String** representation of a number in a specified style to its single-precision floating point number equivalent.

Return Value: A single-precision floating point number equivalent to the numeric value or symbol specified in *s*.

s can contain

System.Globalization.NumberFormatInfo.PositiveInfinitySymbol,

System.Globalization.NumberFormatInfo.NegativeInfinitySymbol,

System.Globalization.NumberFormatInfo.NaNSymbol, or a string of the form:

[ws][sign]integral-digits[.[fractional-digits]][e[sign]exponential-digits][ws]

Optional items are framed in square brackets ('[' and ']'). Items containing the term "digits" consist of a series of numeric characters ranging from 0 to 9. A

System.String containing a number to convert. The combination of one or more

System.Globalization.NumberStyles constants that indicate the permitted format

of *s*.

Parse

```
[C#] public static float Parse(string s, NumberStyles style, IFormatProvider
provider);
[C++] public: static float Parse(String* s, NumberStyles style, IFormatProvider*
provider);
[VB] Public Shared Function Parse(ByVal s As String, ByVal style As
NumberStyles, ByVal provider As IFormatProvider) As Single
[JScript] public static function Parse(s : String, style : NumberStyles, provider :
IFormatProvider) : float;
```

Description

Converts the **String** representation of a number in a specified style and culture-specific format to its single-precision floating point number equivalent.

Return Value: A single-precision floating point number equivalent to the numeric value or symbol specified in *s*.

s can contain

System.Globalization.NumberFormatInfo.PositiveInfinitySymbol ,
System.Globalization.NumberFormatInfo.NegativeInfinitySymbol ,
System.Globalization.NumberFormatInfo.NaNSymbol , or a string of the form:

[ws][sign]integral-digits[.fractional-digits][e[sign]exponential-digits][ws]

Optional items are framed in square brackets ('[' and ']'). Items containing the term "digits" consist of a series of numeric characters ranging from 0 to 9. A

System.String containing a number to convert. The combination of one or more **System.Globalization.NumberStyles** constants that indicate the permitted format

of *s*. An **System.IFormatProvider** interface implementation which supplies culture-specific formatting information about *s*.

Convertible.ToBoolean

[C#] bool Convertible.ToBoolean(IFormatProvider provider);

[C++] bool Convertible::ToBoolean(IFormatProvider* provider);

[VB] Function ToBoolean(ByVal provider As IFormatProvider) As Boolean

Implements Convertible.ToBoolean

[JScript] function Convertible.ToBoolean(provider : IFormatProvider) : Boolean;

Convertible.ToByte

[C#] byte Convertible.ToByte(IFormatProvider provider);

[C++] unsigned char Convertible::ToByte(IFormatProvider* provider);

[VB] Function ToByte(ByVal provider As IFormatProvider) As Byte Implements

Convertible.ToByte

[JScript] function Convertible.ToByte(provider : IFormatProvider) : Byte;

Convertible.ToChar

[C#] char Convertible.ToChar(IFormatProvider provider);

[C++] __wchar_t Convertible::ToChar(IFormatProvider* provider);

[VB] Function ToChar(ByVal provider As IFormatProvider) As Char Implements

Convertible.ToChar

[JScript] function Convertible.ToChar(provider : IFormatProvider) : Char;

Convertible.ToDateTime

```

1
2 [C#] DateTime IConvertible.ToDateTime(IFormatProvider provider);
3 [C++] DateTime IConvertible::ToDateTime(IFormatProvider* provider);
4 [VB] Function ToDateTime(ByVal provider As IFormatProvider) As DateTime
5 Implements IConvertible.ToDateTime
6 [JScript] function IConvertible.ToDateTime(provider : IFormatProvider) :
7 DateTime;
8     IConvertible.ToDecimal
9
10 [C#] decimal IConvertible.ToDecimal(IFormatProvider provider);
11 [C++] Decimal IConvertible::ToDecimal(IFormatProvider* provider);
12 [VB] Function ToDecimal(ByVal provider As IFormatProvider) As Decimal
13 Implements IConvertible.ToDecimal
14 [JScript] function IConvertible.ToDecimal(provider : IFormatProvider) : Decimal;
15     IConvertible.ToDouble
16
17 [C#] double IConvertible.ToDouble(IFormatProvider provider);
18 [C++] double IConvertible::ToDouble(IFormatProvider* provider);
19 [VB] Function ToDouble(ByVal provider As IFormatProvider) As Double
20 Implements IConvertible.ToDouble
21 [JScript] function IConvertible.ToDouble(provider : IFormatProvider) : double;
22     IConvertible.ToInt16
23
24 [C#] short IConvertible.ToInt16(IFormatProvider provider);
25 [C++] short IConvertible::ToInt16(IFormatProvider* provider);

```



```

1 [VB] Function ToInt16(ByVal provider As IFormatProvider) As Short
2 Implements IConvertible.ToInt16
3 [JScript] function IConvertible.ToInt16(provider : IFormatProvider) : Int16;
4     IConvertible.ToInt32
5
6 [C#] int IConvertible.ToInt32(IFormatProvider provider);
7 [C++] int IConvertible::ToInt32(IFormatProvider* provider);
8 [VB] Function ToInt32(ByVal provider As IFormatProvider) As Integer
9 Implements IConvertible.ToInt32
10 [JScript] function IConvertible.ToInt32(provider : IFormatProvider) : int;
11     IConvertible.ToInt64
12
13 [C#] long IConvertible.ToInt64(IFormatProvider provider);
14 [C++] __int64 IConvertible::ToInt64(IFormatProvider* provider);
15 [VB] Function ToInt64(ByVal provider As IFormatProvider) As Long Implements
16 IConvertible.ToInt64
17 [JScript] function IConvertible.ToInt64(provider : IFormatProvider) : long;
18     IConvertible.ToSByte
19
20 [C#] sbyte IConvertible.ToSByte(IFormatProvider provider);
21 [C++] char IConvertible::ToSByte(IFormatProvider* provider);
22 [VB] Function ToSByte(ByVal provider As IFormatProvider) As SByte
23 Implements IConvertible.ToSByte
24 [JScript] function IConvertible.ToSByte(provider : IFormatProvider) : SByte;
25     IConvertible.ToSingle

```

```

1
2 [C#] float IConvertible.ToSingle(IFormatProvider provider);
3 [C++] float IConvertible::ToSingle(IFormatProvider* provider);
4 [VB] Function ToSingle(ByVal provider As IFormatProvider) As Single
5 Implements IConvertible.ToSingle
6 [JScript] function IConvertible.ToSingle(provider : IFormatProvider) : float;
7     IConvertible.ToType
8
9 [C#] object IConvertible.ToType(Type type, IFormatProvider provider);
10 [C++] Object* IConvertible::ToType(Type* type, IFormatProvider* provider);
11 [VB] Function ToType(ByVal type As Type, ByVal provider As IFormatProvider)
12 As Object Implements IConvertible.ToType
13 [JScript] function IConvertible.ToType(type : Type, provider : IFormatProvider) :
14 Object;
15     IConvertible.ToUInt16
16
17 [C#] ushort IConvertible.ToUInt16(IFormatProvider provider);
18 [C++] unsigned short IConvertible::ToUInt16(IFormatProvider* provider);
19 [VB] Function ToUInt16(ByVal provider As IFormatProvider) As UInt16
20 Implements IConvertible.ToUInt16
21 [JScript] function IConvertible.ToUInt16(provider : IFormatProvider) : UInt16;
22     IConvertible.ToUInt32
23
24 [C#] uint IConvertible.ToUInt32(IFormatProvider provider);
25 [C++] unsigned int IConvertible::ToUInt32(IFormatProvider* provider);

```

```

1 [VB] Function ToUInt32(ByVal provider As IFormatProvider) As UInt32
2 Implements IConvertible.ToUInt32
3 [JScript] function IConvertible.ToUInt32(provider : IFormatProvider) : UInt32;
4     IConvertible.ToUInt64
5
6 [C#] ulong IConvertible.ToUInt64(IFormatProvider provider);
7 [C++] unsigned __int64 IConvertible::ToUInt64(IFormatProvider* provider);
8 [VB] Function ToUInt64(ByVal provider As IFormatProvider) As UInt64
9 Implements IConvertible.ToUInt64
10 [JScript] function IConvertible.ToUInt64(provider : IFormatProvider) : UInt64;
11     ToString
12
13 [C#] public override string ToString();
14 [C++] public: String* ToString();
15 [VB] Overrides Public Function ToString() As String
16 [JScript] public override function ToString() : String; Converts the numeric value
17 of this instance to its equivalent String representation.
18

```

Description

Converts the numeric value of this instance to its equivalent **String** representation.

Return Value: The **System.String** representation of the value of this instance.

The return value can be

System.Globalization.NumberFormatInfo.PositiveInfinitySymbol ,
System.Globalization.NumberFormatInfo.NegativeInfinitySymbol ,

System.Globalization.NumberFormatInfo.NaNSymbol , or a string of the form:
[sign]integral-digits[.[fractional-digits]][e[sign]exponential-digits] Optional items
are framed in square brackets ('[' and ']'). Items containing the term "digits" consist
of a series of numeric characters ranging from 0 to 9.

ToString

[C#] public string ToString(IFormatProvider provider);
[C++] public: __sealed String* ToString(IFormatProvider* provider);
[VB] NotOverridable Public Function ToString(ByVal provider As
IFormatProvider) As String
[JScript] public function ToString(provider : IFormatProvider) : String;

Description

Converts the numeric value of this instance to its equivalent **String**
representation using the specified culture-specific format information.

Return Value: The **System.String** representation of the value of this instance as
specified by *provider* .

The return value can be

System.Globalization.NumberFormatInfo.PositiveInfinitySymbol ,
System.Globalization.NumberFormatInfo.NegativeInfinitySymbol ,
System.Globalization.NumberFormatInfo.NaNSymbol , or a string of the form:
[sign]integral-digits[.[fractional-digits]][e[sign]exponential-digits] Optional items
are framed in square brackets ('[' and ']'). Items containing the term "digits" consist
of a series of numeric characters ranging from 0 to 9. An

System.IFormatProvider interface implementation which supplies culture-specific formatting information.

ToString

[C#] public string ToString(string format);

[C++] public: String* ToString(String* format);

[VB] Public Function ToString(ByVal format As String) As String

[JScript] public function ToString(format : String) : String;

Description

Converts the numeric value of this instance to its equivalent **String** representation, using the specified format.

Return Value: The **System.String** representation of the value of this instance as specified by *format* .

The return value can be

System.Globalization.NumberFormatInfo.PositiveInfinitySymbol ,

System.Globalization.NumberFormatInfo.NegativeInfinitySymbol ,

System.Globalization.NumberFormatInfo.NaNSymbol , or a string of the form:

[sign]integral-digits[.[fractional-digits]][e[sign]exponential-digits] Optional items are framed in square brackets ('[' and ']'). Items containing the term "digits" consist of a series of numeric characters ranging from 0 to 9. A format string.

ToString

[C#] public string ToString(string format, IFormatProvider provider);

[C++] public: __sealed String* ToString(String* format, IFormatProvider*

provider);

[VB] NotOverridable Public Function ToString(ByVal format As String, ByVal provider As IFormatProvider) As String

[JScript] public function ToString(format : String, provider : IFormatProvider) : String;

Description

Converts the numeric value of this instance to its equivalent **String** representation using the specified format and culture-specific format information.

Return Value: The **System.String** representation of the value of this instance as specified by *format* and *provider* .

The return value can be

System.Globalization.NumberFormatInfo.PositiveInfinitySymbol ,

System.Globalization.NumberFormatInfo.NegativeInfinitySymbol ,

System.Globalization.NumberFormatInfo.NaNSymbol , or a string of the form:

[sign]integral-digits[.fractional-digits][e[sign]exponential-digits] Optional items

are framed in square brackets ('[' and ']'). Items containing the term "digits" consist

of a series of numeric characters ranging from 0 to 9. A format specification. An

System.IFormatProvider interface implementation which supplies culture-specific formatting information.

Environment.SpecialFolder enumeration (System)

ToString

Description

Specifies enumerated constants used to retrieve directory paths to system special folders.

The **System.Environment.SpecialFolder.System** method uses these enumerated constants to indicate the special folder path to retrieve.

ToString

[C#] public const Environment.SpecialFolder ApplicationData;

[C++] public: const Environment.SpecialFolder ApplicationData;

[VB] Public Const ApplicationData As Environment.SpecialFolder

[JScript] public var ApplicationData : Environment.SpecialFolder;

Description

The directory that serves as a common repository for application-specific data for the current, roaming user.

ToString

[C#] public const Environment.SpecialFolder CommonApplicationData;

[C++] public: const Environment.SpecialFolder CommonApplicationData;

[VB] Public Const CommonApplicationData As Environment.SpecialFolder

[JScript] public var CommonApplicationData : Environment.SpecialFolder;

Description

The directory that serves as a common repository for application-specific data that is used by all users.

ToString

1
2 [C#] public const Environment.SpecialFolder CommonProgramFiles;

3 [C++] public: const Environment.SpecialFolder CommonProgramFiles;

4 [VB] Public Const CommonProgramFiles As Environment.SpecialFolder

5 [JScript] public var CommonProgramFiles : Environment.SpecialFolder;

6
7 *Description*

8 The directory for components that are shared across applications.

9 ToString

10
11 [C#] public const Environment.SpecialFolder Cookies;

12 [C++] public: const Environment.SpecialFolder Cookies;

13 [VB] Public Const Cookies As Environment.SpecialFolder

14 [JScript] public var Cookies : Environment.SpecialFolder;

15
16 *Description*

17 The directory that serves as a common repository for Internet cookies.

18 ToString

19
20 [C#] public const Environment.SpecialFolder DesktopDirectory;

21 [C++] public: const Environment.SpecialFolder DesktopDirectory;

22 [VB] Public Const DesktopDirectory As Environment.SpecialFolder

23 [JScript] public var DesktopDirectory : Environment.SpecialFolder;

24
25 *Description*

The directory used to physically store file objects on the desktop.

ToString

[C#] public const Environment.SpecialFolder Favorites;

[C++] public: const Environment.SpecialFolder Favorites;

[VB] Public Const Favorites As Environment.SpecialFolder

[JScript] public var Favorites : Environment.SpecialFolder;

Description

The directory that serves as a common repository for the user's favorite items.

ToString

[C#] public const Environment.SpecialFolder History;

[C++] public: const Environment.SpecialFolder History;

[VB] Public Const History As Environment.SpecialFolder

[JScript] public var History : Environment.SpecialFolder;

Description

The directory that serves as a common repository for Internet history items.

ToString

[C#] public const Environment.SpecialFolder InternetCache;

[C++] public: const Environment.SpecialFolder InternetCache;

[VB] Public Const InternetCache As Environment.SpecialFolder

1 [JScript] public var InternetCache : Environment.SpecialFolder;

3 *Description*

4 The directory that serves as a common repository for temporary Internet
5 files.

6 ToString

8 [C#] public const Environment.SpecialFolder LocalApplicationData;

9 [C++] public: const Environment.SpecialFolder LocalApplicationData;

10 [VB] Public Const LocalApplicationData As Environment.SpecialFolder

11 [JScript] public var LocalApplicationData : Environment.SpecialFolder;

13 *Description*

14 The directory that serves as a common repository for application-specific
15 data that is used by the current, non-roaming user.

16 ToString

18 [C#] public const Environment.SpecialFolder Personal;

19 [C++] public: const Environment.SpecialFolder Personal;

20 [VB] Public Const Personal As Environment.SpecialFolder

21 [JScript] public var Personal : Environment.SpecialFolder;

23 *Description*

24 The directory that serves as a common repository for documents.

25 ToString

```

1
2 [C#] public const Environment.SpecialFolder ProgramFiles;
3 [C++] public: const Environment.SpecialFolder ProgramFiles;
4 [VB] Public Const ProgramFiles As Environment.SpecialFolder
5 [JScript] public var ProgramFiles : Environment.SpecialFolder;
6

```

Description

The program files directory.

ToString

```

11 [C#] public const Environment.SpecialFolder Programs;
12 [C++] public: const Environment.SpecialFolder Programs;
13 [VB] Public Const Programs As Environment.SpecialFolder
14 [JScript] public var Programs : Environment.SpecialFolder;
15

```

Description

The directory that contains the user's program groups.

ToString

```

20 [C#] public const Environment.SpecialFolder Recent;
21 [C++] public: const Environment.SpecialFolder Recent;
22 [VB] Public Const Recent As Environment.SpecialFolder
23 [JScript] public var Recent : Environment.SpecialFolder;
24

```

Description

The directory that contains the user's most recently used documents.

ToString

[C#] public const Environment.SpecialFolder SendTo;

[C++] public: const Environment.SpecialFolder SendTo;

[VB] Public Const SendTo As Environment.SpecialFolder

[JScript] public var SendTo : Environment.SpecialFolder;

Description

The directory that contains Send To menu items.

ToString

[C#] public const Environment.SpecialFolder StartMenu;

[C++] public: const Environment.SpecialFolder StartMenu;

[VB] Public Const StartMenu As Environment.SpecialFolder

[JScript] public var StartMenu : Environment.SpecialFolder;

Description

The directory that contains the Start menu items.

ToString

[C#] public const Environment.SpecialFolder Startup;

[C++] public: const Environment.SpecialFolder Startup;

[VB] Public Const Startup As Environment.SpecialFolder

[JScript] public var Startup : Environment.SpecialFolder;

1
2 *Description*

3 The directory that corresponds to the user's Startup program group.

4 ToString

5
6 [C#] public const Environment.SpecialFolder System;

7 [C++] public: const Environment.SpecialFolder System;

8 [VB] Public Const System As Environment.SpecialFolder

9 [JScript] public var System : Environment.SpecialFolder;

10
11 *Description*

12 The System directory.

13 ToString

14
15 [C#] public const Environment.SpecialFolder Templates;

16 [C++] public: const Environment.SpecialFolder Templates;

17 [VB] Public Const Templates As Environment.SpecialFolder

18 [JScript] public var Templates : Environment.SpecialFolder;

19
20 *Description*

21 The directory that serves as a common repository for document templates.

22 StackOverflowException class (System)

23 ToString

Description

The exception that is thrown when the execution stack overflows by having too many pending method calls. This class cannot be inherited.

System.StackOverflowException is thrown for execution stack overflow errors, typically in case of a very deep or unbounded recursion.

StackOverflowException

Example Syntax:

ToString

[C#] public StackOverflowException();

[C++] public: StackOverflowException();

[VB] Public Sub New()

[JScript] public function StackOverflowException(); Initializes a new instance of the **System.StackOverflowException** class.

Description

Initializes a new instance of the **System.StackOverflowException** class with default properties.

The following table shows the initial property values for an instance of **System.StackOverflowException**.

StackOverflowException

Example Syntax:

ToString

```

1
2 [C#] public StackOverflowException(string message);
3 [C++] public: StackOverflowException(String* message);
4 [VB] Public Sub New(ByVal message As String)
5 [JScript] public function StackOverflowException(message : String);
6

```

Description

Initializes a new instance of the **System.StackOverflowException** class with a specified error message.

The following table shows the initial property values for an instance of **System.StackOverflowException**. The error message that explains the reason for the exception.

Property	Initial Value
StackOverflowException	

Example Syntax:

ToString

```

17 [C#] public StackOverflowException(string message, Exception innerException);
18 [C++] public: StackOverflowException(String* message, Exception*
19 innerException);
20 [VB] Public Sub New(ByVal message As String, ByVal innerException As
21 Exception)
22 [JScript] public function StackOverflowException(message : String,
23 innerException : Exception);
24

```

Description

1 Initializes a new instance of the **System.StackOverflowException** class
2 with a specified error message and a reference to the inner exception that is the
3 root cause of this exception.

4 When an **Exception** *X* is thrown as a direct result of a previous exception *Y* ,
5 the **System.Exception.InnerException** property of *X* should contain a reference
6 to *Y* . The **InnerException** property returns the same value as was passed into the
7 constructor, or **null** if the inner exception value was not supplied to the
8 constructor. The error message that explains the reason for the exception. An
9 instance of **System.Exception** that is the cause of the current **Exception**. If
10 *innerException* is non-null, then the current **Exception** is raised in a catch block
11 handling *innerException* .

12 HelpLink

13 HResult

14 InnerException

15 Message

16 Source

17 StackTrace

18 TargetSite

19 STAThreadAttribute class (System)

20 ToString

21
22
23 *Description*

24 Indicates the default threading model for an application is single-threaded
25 apartment.

Only apply this attribute to the main method of an application.

STAThreadAttribute

Example Syntax:

ToString

[C#] public STAThreadAttribute();

[C++] public: STAThreadAttribute();

[VB] Public Sub New()

[JScript] public function STAThreadAttribute();

Description

Initializes a new instance of the **System.STAThreadAttribute** class.

TypeId

String class (System)

ToString

Description

Represents an immutable string of characters.

An instance of **String** is said to be "immutable" because its value cannot be modified once it has been created. Methods that appear to modify a **String** instance actually return a new instance containing the modification. Use the **System.Text.StringBuilder** class if it is necessary to actually modify the contents of a string-like object.

ToString

[C#] public static readonly string Empty;

[C++] public: static String* Empty;

[VB] Public Shared ReadOnly Empty As String

[JScript] public static var Empty : String;

Description

A read-only field that represents the empty string.

The value of this field is the string, "".

String

Example Syntax:

ToString

[C#] unsafe public String(char* value);

[C++] public: String(__wchar_t* value); Initializes a new instance of the **String** class.

Description

Initializes a new instance of the **String** class to the value indicated by a specified pointer to an array of Unicode characters.

If *value* is a null pointer, an **System.String.Empty** instance is initialized. A pointer to an array of Unicode characters.

String

Example Syntax:

ToString

[C#] public String(char[] value);
[C++] public: String(__wchar_t value __gc[]);
[VB] Public Sub New(ByVal value() As Char)
[JScript] public function String(value : Char[]);

Description

Initializes a new instance of the **String** class to the value indicated by an array of Unicode characters. An array of Unicode characters.

String

Example Syntax:

ToString

[C#] unsafe public String(sbyte* value);
[C++] public: String(char* value);

Description

Initializes a new instance of the **String** class to the value indicated by a pointer to an array of 8-bit signed integers.

If *value* is a null pointer, an **System.String.Empty** instance is initialized. A pointer to an array of 8-bit signed integers.

String

Example Syntax:

ToString

1
2 [C#] public String(char c, int count);

3 [C++] public: String(__wchar_t c, int count);

4 [VB] Public Sub New(ByVal c As Char, ByVal count As Integer)

5 [JScript] public function String(c : Char, count : int);

6
7 *Description*

8 Initializes a new instance of the **String** class to the value indicated by a
9 specified Unicode character repeated a specified number of times. A Unicode
10 character. The number of times *c* occurs.

11 String

12 *Example Syntax:*

13 ToString

14
15 [C#] unsafe public String(char* value, int startIndex, int length);

16 [C++] public: String(__wchar_t* value, int startIndex, int length);

17
18 *Description*

19 Initializes a new instance of the **String** class to the value indicated by a
20 specified pointer to an array of Unicode characters, a starting character position
21 within that array, and a length.

22 If *value* is a null pointer, an **System.String.Empty** instance is initialized. A
23 pointer to an array of Unicode characters. The starting position within *value*. The
24 number of characters within *value* to use.

25 String

Example Syntax:

ToString

[C#] public String(char[] value, int startIndex, int length);

[C++] public: String(__wchar_t value __gc[], int startIndex, int length);

[VB] Public Sub New(ByVal value() As Char, ByVal startIndex As Integer,
ByVal length As Integer)

[JScript] public function String(value : Char[], startIndex : int, length : int);

Description

Initializes a new instance of the **String** class to the value indicated by an array of Unicode characters, a starting character position within that array, and a length.

If *value* is **null**, an **System.String.Empty** instance is initialized. An array of Unicode characters. The starting position within *value*. The number of characters within *value* to use.

String

Example Syntax:

ToString

[C#] unsafe public String(sbyte* value, int startIndex, int length);

[C++] public: String(char* value, int startIndex, int length);

Description

1 Initializes a new instance of the **String** class to the value indicated by a
2 specified pointer to an array of 8-bit signed integers, a starting character position
3 within that array, and a length.

4 If *value* is a null pointer, an **System.String.Empty** instance is initialized. A
5 pointer to an array of 8-bit signed integers. The starting position within *value*. The
6 number of characters within *value* to use.

7 **String**

8 *Example Syntax:*

9 **ToString**

10
11 [C#] unsafe public String(sbyte* value, int startIndex, int length, Encoding enc);

12 [C++] public: String(char* value, int startIndex, int length, Encoding* enc);

13
14 *Description*

15 Initializes a new instance of the **String** class to the value indicated by a
16 specified pointer to an array of 8-bit signed integers, a starting character position
17 within that array, a length, and an **Encoding** object.

18 If *value* is a null pointer, an **System.String.Empty** instance is initialized. A
19 pointer to an array of 8-bit signed integers. The starting position within *value*. The
20 number of characters within *value* to use. An **System.Text.Encoding** object that
21 specifies how the array referenced by *value* is encoded.

22 **Chars**

23 **ToString**

24
25 [C#] public char this[int index] {get;}

1 [C++] public: __property __wchar_t get_Chars(int index);

2 [VB] Public Default ReadOnly Property Chars(ByVal index As Integer) As Char

3 [JScript] returnValue = StringObject.Chars(index);

4
5 *Description*

6 Gets the character at a specified character position in this instance.

7 *index* is zero-based. A character position in this instance.

8 Length

9 ToString

10
11 [C#] public int Length {get;}

12 [C++] public: __property int get_Length();

13 [VB] Public ReadOnly Property Length As Integer

14 [JScript] public function get Length() : int;

15
16 *Description*

17 Gets the number of characters in this instance.

18 Clone

19
20 [C#] public object Clone();

21 [C++] public: __sealed Object* Clone();

22 [VB] NotOverridable Public Function Clone() As Object

23 [JScript] public function Clone() : Object;

24
25 *Description*

Returns a reference to this instance of **String** .

Return Value: This instance of **String** .

The return value is not an independent copy of this instance; it is simply another view of the same data. Use the **System.String.Copy(System.String)** or **System.String.CopyTo(System.Int32,System.Char[],System.Int32,System.Int32)** method to create a separate **String** object with the same value as this instance.

Compare

[C#] public static int Compare(string strA, string strB);

[C++] public: static int Compare(String* strA, String* strB);

[VB] Public Shared Function Compare(ByVal strA As String, ByVal strB As String) As Integer

[JScript] public static function Compare(strA : String, strB : String) : int;

Compares two specified **String** objects.

Description

Compares two specified **String** objects.

Return Value: A 32-bit signed integer indicating the lexical relationship between the two comparands.

By definition, any **String** , including the empty string, compares greater than a null reference; and two null references compare equal to each other. The first **String**. The second **String**.

Compare

[C#] public static int Compare(string strA, string strB, bool ignoreCase);


```

1 [C++] public: static int Compare(String* strA, String* strB, bool ignoreCase);
2 [VB] Public Shared Function Compare(ByVal strA As String, ByVal strB As
3 String, ByVal ignoreCase As Boolean) As Integer
4 [JScript] public static function Compare(strA : String, strB : String, ignoreCase :
5 Boolean) : int;

```

Description

Compares two specified **String** objects, ignoring or honoring their case.

Return Value: A 32-bit signed integer indicating the lexical relationship between the two comparands.

By definition, any **String**, including the empty string, compares greater than a null reference; and two null references compare equal to each other. The first **String**. The second **String**. A **System.Boolean** indicating a case-sensitive or insensitive comparison. (**true** indicates a case-insensitive comparison.)

Compare

```

17 [C#] public static int Compare(string strA, string strB, bool ignoreCase,
18 CultureInfo culture);
19 [C++] public: static int Compare(String* strA, String* strB, bool ignoreCase,
20 CultureInfo* culture);
21 [VB] Public Shared Function Compare(ByVal strA As String, ByVal strB As
22 String, ByVal ignoreCase As Boolean, ByVal culture As CultureInfo) As Integer
23 [JScript] public static function Compare(strA : String, strB : String, ignoreCase :
24 Boolean, culture : CultureInfo) : int;

```

Description

Compares two specified **String** objects, ignoring or honoring their case, and honoring culture-specific information about their formatting.

Return Value: A 32-bit signed integer indicating the lexical relationship between the two comparands.

culture specifies a **System.Globalization.CultureInfo** object, which provides culture-specific information that can affect the comparison. The first **String**. The second **String**. A **System.Boolean** indicating a case-sensitive or insensitive comparison. (**true** indicates a case-insensitive comparison.) A **System.Globalization.CultureInfo** object that supplies culture-specific formatting information.

Compare

[C#] public static int Compare(string strA, int indexA, string strB, int indexB, int length);

[C++] public: static int Compare(String* strA, int indexA, String* strB, int indexB, int length);

[VB] Public Shared Function Compare(ByVal strA As String, ByVal indexA As Integer, ByVal strB As String, ByVal indexB As Integer, ByVal length As Integer) As Integer

[JScript] public static function Compare(strA : String, indexA : int, strB : String, indexB : int, length : int) : int;

Description

Compares substrings of two specified **String** objects.

Return Value: A 32-bit signed integer indicating the lexical relationship between the two comparands.

length cannot be negative. If *length* is zero, then zero is returned. The first **String**. The position of the substring within *strA*. The second **String**. The position of the substring within *strB*. The maximum number of characters in the substrings to compare.

Compare

```
[C#] public static int Compare(string strA, int indexA, string strB, int indexB, int length, bool ignoreCase);
```

```
[C++] public: static int Compare(String* strA, int indexA, String* strB, int indexB, int length, bool ignoreCase);
```

```
[VB] Public Shared Function Compare(ByVal strA As String, ByVal indexA As Integer, ByVal strB As String, ByVal indexB As Integer, ByVal length As Integer, ByVal ignoreCase As Boolean) As Integer
```

```
[JScript] public static function Compare(strA : String, indexA : int, strB : String, indexB : int, length : int, ignoreCase : Boolean) : int;
```

Description

Compares substrings of two specified **String** objects, ignoring or honoring their case.

Return Value: A 32-bit signed integer indicating the lexical relationship between the two comparands.

indexA and *indexB* are zero-based. The first **String**. The position of the substring within *strA*. The second **String**. The position of the substring within *strB*. The maximum number of characters in the substrings to compare. A **System.Boolean** indicating a case-sensitive or insensitive comparison. (**true** indicates a case-insensitive comparison.)

Compare

[C#] public static int Compare(string strA, int indexA, string strB, int indexB, int length, bool ignoreCase, CultureInfo culture);

[C++] public: static int Compare(String* strA, int indexA, String* strB, int indexB, int length, bool ignoreCase, CultureInfo* culture);

[VB] Public Shared Function Compare(ByVal strA As String, ByVal indexA As Integer, ByVal strB As String, ByVal indexB As Integer, ByVal length As Integer, ByVal ignoreCase As Boolean, ByVal culture As CultureInfo) As Integer

[JScript] public static function Compare(strA : String, indexA : int, strB : String, indexB : int, length : int, ignoreCase : Boolean, culture : CultureInfo) : int;

Description

Compares substrings of two specified **String** objects, ignoring or honoring their case, and honoring culture-specific information about their formatting.

Return Value: An integer indicating the lexical relationship between the two comparands.

culture specifies a **System.Globalization.CultureInfo** object, which provides culture-specific information that can affect the comparison. The first **String**. The position of the substring within *strA*. The second **String**. The position

of the substring within the *strB*. The maximum number of characters in the substrings to compare. A **System.Boolean** indicating a case-sensitive or insensitive comparison. (**true** indicates a case-insensitive comparison.) A **System.Globalization.CultureInfo** object that supplies culture-specific formatting information.

CompareOrdinal

[C#] public static int CompareOrdinal(string strA, string strB);

[C++] public: static int CompareOrdinal(String* strA, String* strB);

[VB] Public Shared Function CompareOrdinal(ByVal strA As String, ByVal strB As String) As Integer

[JScript] public static function CompareOrdinal(strA : String, strB : String) : int;

Compares two **String** objects, without considering the local national language or culture.

Description

Compares two specified **String** objects, without considering the local national language or culture.

Return Value: An integer indicating the lexical relationship between the two comparands.

By definition, any **String**, including the empty string, compares greater than a null reference; and two null references compare equal to each other. The first **String**. The second **String**.

CompareOrdinal

```

1
2 [C#] public static int CompareOrdinal(string strA, int indexA, string strB, int
3 indexB, int length);
4 [C++] public: static int CompareOrdinal(String* strA, int indexA, String* strB, int
5 indexB, int length);
6 [VB] Public Shared Function CompareOrdinal(ByVal strA As String, ByVal
7 indexA As Integer, ByVal strB As String, ByVal indexB As Integer, ByVal length
8 As Integer) As Integer
9 [JScript] public static function CompareOrdinal(strA : String, indexA : int, strB :
10 String, indexB : int, length : int) : int;
11

```

Description

Compares substrings of two specified **String** objects, without considering the local national language or culture. Parameters specify the length and starting positions of the substrings.

Return Value: A 32-bit signed integer indicating the lexical relationship between the two comparands.

By definition, any **String**, including the empty string, compares greater than a null reference; and two null references compare equal to each other. The first **String**. The starting index of the substring in *strA*. The second **String**. The starting index of the substring in *strB*. The maximum number of characters in the substrings to compare.

CompareTo

```

23
24
25 [C#] public int CompareTo(object value);

```

1 [C++] public: __sealed int CompareTo(Object* value);

2 [VB] NotOverridable Public Function CompareTo(ByVal value As Object) As

3 Integer

4 [JScript] public function CompareTo(value : Object) : int; Compares this instance
5 with a specified object.

6
7 *Description*

8 Compares this instance with a specified **Object** .

9 *Return Value:* A 32-bit signed integer indicating the lexical relationship between
10 the two comparands.

11 *value* must be a **String** object. An **System.Object** that evaluates to a

12 **String**.

13 CompareTo

14
15 [C#] public int CompareTo(string strB);

16 [C++] public: int CompareTo(String* strB);

17 [VB] Public Function CompareTo(ByVal strB As String) As Integer

18 [JScript] public function CompareTo(strB : String) : int;

19
20 *Description*

21 Compares this instance with a specified **String** object.

22 *Return Value:* A 32-bit signed integer indicating the lexical relationship between
23 the two comparands.

By definition, any **String** , including the empty string, compares greater than a null reference; and two null references compare equal to each other. A

String.

Concat

[C#] public static string Concat(object arg0);

[C++] public: static String* Concat(Object* arg0);

[VB] Public Shared Function Concat(ByVal arg0 As Object) As String

[JScript] public static function Concat(arg0 : Object) : String; Concatenates one or more instances of **String** , or the **String** representations of the values of one or more instances of **Object** .

Description

Creates the **String** representation of a specified object.

Return Value: The **String** representation of the value of *arg0* .

An **System.String.Empty** string is used in place of any null argument. An **System.Object** or **null**.

Concat

[C#] public static string Concat(params object[] args);

[C++] public: static String* Concat(Object* args __gc[]);

[VB] Public Shared Function Concat(ByVal ParamArray args() As Object) As String

[JScript] public static function Concat(args : Object[]) : String;

Description

Concatenates the **String** representations of the elements in a specified

Object array.

Return Value: The concatenated **String** representations of the values of the elements in *args* .

An **System.String.Empty** string is used in place of any null object in the array. An **System.Object** array.

Concat

[C#] public static string Concat(params string[] values);

[C++] public: static String* Concat(String* values __gc[]);

[VB] Public Shared Function Concat(ByVal ParamArray values() As String) As

String

[JScript] public static function Concat(values : String[]) : String;

Description

Concatenates the elements of a specified **String** array.

Return Value: The concatenated elements of *values* .

An **System.String.Empty** string is used in place of any null object in the array. An array of **String** instances.

Concat

[C#] public static string Concat(object arg0, object arg1);

[C++] public: static String* Concat(Object* arg0, Object* arg1);

1 [VB] Public Shared Function Concat(ByVal arg0 As Object, ByVal arg1 As
2 Object) As String

3 [JScript] public static function Concat(arg0 : Object, arg1 : Object) : String;

4
5 *Description*

6 Concatenates the **String** representations of two specified objects.

7 *Return Value:* The concatenated **String** representations of the values of *arg0* and
8 *arg1* .

9 An **System.String.Empty** string is used in place of any null argument. The
10 first **System.Object**. The second **Object**.

11 Concat

12
13 [C#] public static string Concat(string str0, string str1);

14 [C++] public: static String* Concat(String* str0, String* str1);

15 [VB] Public Shared Function Concat(ByVal str0 As String, ByVal str1 As String)
16 As String

17 [JScript] public static function Concat(str0 : String, str1 : String) : String;

18
19 *Description*

20 Concatenates two specified instances of **String** .

21 *Return Value:* The concatenation of *str0* and *str1* .

22 An **System.String.Empty** string is used in place of any null argument. The
23 first **String**. The second **String**.

24 Concat

1
2 [C#] public static string Concat(object arg0, object arg1, object arg2);

3 [C++] public: static String* Concat(Object* arg0, Object* arg1, Object* arg2);

4 [VB] Public Shared Function Concat(ByVal arg0 As Object, ByVal arg1 As
5 Object, ByVal arg2 As Object) As String

6 [JScript] public static function Concat(arg0 : Object, arg1 : Object, arg2 : Object) :
7 String;

8
9 *Description*

10 Concatenates the **String** representations of three specified objects.

11 *Return Value:* The concatenated **String** representations of the values of *arg0* ,
12 *arg1* , and *arg2* .

13 An **System.String.Empty** string is used in place of any null argument. The
14 first **System.Object**. The second **Object**. The third **Object**.

15 **Concat**

16
17 [C#] public static string Concat(string str0, string str1, string str2);

18 [C++] public: static String* Concat(String* str0, String* str1, String* str2);

19 [VB] Public Shared Function Concat(ByVal str0 As String, ByVal str1 As String,
20 ByVal str2 As String) As String

21 [JScript] public static function Concat(str0 : String, str1 : String, str2 : String) :
22 String;

23
24 *Description*

Concatenates three specified instances of **String** .

Return Value: The concatenation of *str0* , *str1* , and *str2* .

An **System.String.Empty** string is used in place of any null argument. The first **String**. The second **String**. The third **String**.

Concat

```
[C++] public: static String* Concat(Object* arg0, Object* arg1, Object* arg2,
Object* arg3, ...);
```

Concat

```
[C#] public static string Concat(string str0, string str1, string str2, string str3);
```

```
[C++] public: static String* Concat(String* str0, String* str1, String* str2, String*
str3);
```

```
[VB] Public Shared Function Concat(ByVal str0 As String, ByVal str1 As String,
ByVal str2 As String, ByVal str3 As String) As String
```

```
[JScript] public static function Concat(str0 : String, str1 : String, str2 : String, str3
: String) : String;
```

Description

Concatenates four specified instances of **String** .

Return Value: The concatenation of *str0* , *str1* , *str2*, and *str3* .

An **System.String.Empty** string is used in place of any null object in the array. The first **String**. The second **String**. The third **String**. The fourth **String**.

Copy

1
2 [C#] public static string Copy(string str);

3 [C++] public: static String* Copy(String* str);

4 [VB] Public Shared Function Copy(ByVal str As String) As String

5 [JScript] public static function Copy(str : String) : String;

6
7 *Description*

8 Creates a new instance of **String** with the same value as a specified
9 instance of **String** .

10 *Return Value:* A new **String** with the same value as *str* . The **String** to be copied

11 CopyTo

12
13 [C#] public void CopyTo(int sourceIndex, char[] destination, int destinationIndex,
14 int count);

15 [C++] public: void CopyTo(int sourceIndex, __wchar_t destination __gc[], int
16 destinationIndex, int count);

17 [VB] Public Sub CopyTo(ByVal sourceIndex As Integer, ByVal destination() As
18 Char, ByVal destinationIndex As Integer, ByVal count As Integer)

19 [JScript] public function CopyTo(sourceIndex : int, destination : Char[],
20 destinationIndex : int, count : int);

21
22 *Description*

23 Copies a specified number of characters from a specified position in this
24 instance to a specified position in an array of Unicode characters.

1 *count* characters are copied from the *sourceIndex* position of this instance
2 to the *destinationIndex* position of *destination* . A character position in this
3 instance. An array of Unicode characters. An array element in *destination*. The
4 number of characters in this instance to copy to *destination*.

5 EndsWith

6
7 [C#] public bool EndsWith(string value);

8 [C++] public: bool EndsWith(String* value);

9 [VB] Public Function EndsWith(ByVal value As String) As Boolean

10 [JScript] public function EndsWith(value : String) : Boolean;

11 12 *Description*

13 Determines whether the end of this instance matches the specified **String** .

14 *Return Value:* **true** if the end of this instance matches *value* ; **false** if *value* does
15 not match or is longer than this instance.

16 The comparison is case-sensitive. A **String**.

17 Equals

18
19 [C#] public override bool Equals(object obj);

20 [C++] public: bool Equals(Object* obj);

21 [VB] Overrides Public Function Equals(ByVal obj As Object) As Boolean

22 [JScript] public override function Equals(obj : Object) : Boolean; Determines
23 whether two **String** objects have the same value.

24 25 *Description*

Determines whether this instance of **String** and a specified object, which must be a **String** , have the same value.

Return Value: **true** if *obj* is a **String** and its value is the same as this instance; otherwise, **false**. This instance is **null** .

This comparison is case-sensitive. An **System.Object**.

Equals

[C#] public new bool Equals(string value);

[C++] public: bool Equals(String* value);

[VB] Shadows Public Function Equals(ByVal value As String) As Boolean

[JScript] public hide function Equals(value : String) : Boolean;

Description

Determines whether this instance and a specified **String** have the same value.

Return Value: **true** if the value of *value* is the same as this instance; otherwise, **false**. This instance is **null** .

This comparison is case-sensitive. A **String**.

Equals

[C#] public static new bool Equals(string a, string b);

[C++] public: static bool Equals(String* a, String* b);

[VB] Shadows Public Shared Function Equals(ByVal a As String, ByVal b As String) As Boolean

[JScript] public static hide function Equals(a : String, b : String) : Boolean;

Description

Determines whether two specified **String** objects have the same value.

Return Value: **true** if the value of *a* is the same as the value of *b* ; otherwise, **false**.

The comparison is case-sensitive. A **String** or **null**. A **String** or **null**.

Format

[C#] public static string Format(string format, object arg0);

[C++] public: static String* Format(String* format, Object* arg0);

[VB] Public Shared Function Format(ByVal format As String, ByVal arg0 As Object) As String

[JScript] public static function Format(format : String, arg0 : Object) : String;

Replaces each format specification in a specified **String** with the textual equivalent of a corresponding object's value.

Description

Replaces the format specification in a specified **String** with the textual equivalent of the value of a specified **Object** instance.

Return Value: A copy of *format* in which the first format specification has been replaced by the **String** equivalent of the *arg0* .

The *format* parameter is embedded with zero or more format specifications of the form, { *N* [, *M*][:*formatString*]} , where: *N* is a zero-based integer indicating the argument to be formatted. A **String** containing zero or more format specifications. An **System.Object** to be formatted.

Format


```

1
2 [C#] public static string Format(string format, params object[] args);
3 [C++] public: static String* Format(String* format, Object* args __gc[]);
4 [VB] Public Shared Function Format(ByVal format As String, ByVal ParamArray
5 args() As Object) As String
6 [JScript] public static function Format(format : String, args : Object[]) : String;
7

```

Description

Replaces the format specification in a specified **String** with the textual equivalent of the value of a corresponding **Object** instance in a specified array.

Return Value: A copy of *format* in which the format specifications have been replaced by the **String** equivalent of the corresponding instances of **Object** in *args*

.

The *format* parameter is embedded with zero or more format specifications of the form, { *N* [, *M*][: *formatString*]} , where: *N* is a zero-based integer indicating the argument to be formatted. A **String** containing zero or more format specifications. An **System.Object** array containing zero or more objects to be formatted.

Format

```

21 [C#] public static string Format(IFormatProvider provider, string format, params
22 object[] args);
23 [C++] public: static String* Format(IFormatProvider* provider, String* format,
24 Object* args __gc[]);
25 [VB] Public Shared Function Format(ByVal provider As IFormatProvider, ByVal

```

1 format As String, ByVal ParamArray args() As Object) As String

2 [JScript] public static function Format(provider : IFormatProvider, format : String,
3 args : Object[]) : String;

5 *Description*

6 Replaces the format specification in a specified **String** with the textual
7 equivalent of the value of a corresponding **Object** instance in a specified array. A
8 specified parameter supplies culture-specific formatting information.

9 *Return Value:* A copy of *format* in which the format specifications have been
10 replaced by the **String** equivalent of the corresponding instances of **Object** in *args*
11 .

12 The *format* parameter is embedded with zero or more format specifications
13 of the form, { *N* [, *M*][: *formatString*]} , where: *N* is a zero-based integer
14 indicating the argument to be formatted. An **System.IFormatProvider** interface
15 implementation that supplies culture-specific formatting information. A **String**
16 containing zero or more format specifications. An **System.Object** array containing
17 zero or more objects to be formatted.

18 **Format**

19
20 [C#] public static string Format(string format, object arg0, object arg1);

21 [C++] public: static String* Format(String* format, Object* arg0, Object* arg1);

22 [VB] Public Shared Function Format(ByVal format As String, ByVal arg0 As
23 Object, ByVal arg1 As Object) As String

24 [JScript] public static function Format(format : String, arg0 : Object, arg1 :
25 Object) : String;

Description

Replaces the format specification in a specified **String** with the textual equivalent of the value of two specified **Object** instances.

Return Value: A copy of *format* in which the first and second format specifications have been replaced by the **String** equivalent of the *arg0* and *arg1* .

A **String** containing zero or more format specifications. The first **System.Object** to be formatted. The second **Object** to be formatted.

Format

[C#] public static string Format(string format, object arg0, object arg1, object arg2);

[C++] public: static String* Format(String* format, Object* arg0, Object* arg1, Object* arg2);

[VB] Public Shared Function Format(ByVal format As String, ByVal arg0 As Object, ByVal arg1 As Object, ByVal arg2 As Object) As String

[JScript] public static function Format(format : String, arg0 : Object, arg1 : Object, arg2 : Object) : String;

Description

Replaces the format specification in a specified **String** with the textual equivalent of the value of three specified **Object** instances.

Return Value: A copy of *format* in which the first, second, and third format specifications have been replaced by the **String** equivalent of the *arg0*, *arg1*, and *arg2* .

The *format* parameter is embedded with zero or more format specifications of the form, { *N* [, *M*][: *formatString*]} , where: *N* is a zero-based integer indicating the argument to be formatted. A **String** containing zero or more format specifications. The first **System.Object** to be formatted. The second **Object** to be formatted. The third **Object** to be formatted.

GetEnumerator

```
[C#] public CharEnumerator GetEnumerator();  
[C++] public: CharEnumerator* GetEnumerator();  
[VB] Public Function GetEnumerator() As CharEnumerator  
[JScript] public function GetEnumerator() : CharEnumerator;
```

Description

Retrieves an object that can iterate through the individual characters in this instance.

Return Value: A **System.CharEnumerator** object.

This method is required by programming languages that support the **System.Collections.IEnumerator** interface to iterate through members of a collection. For example, the Microsoft Visual Basic and C# programming languages' **foreach** statement invokes this method to return a **CharEnumerator** object that can provide read-only access to the characters in this instance of **String**.

GetHashCode

```
[C#] public override int GetHashCode();
```

1 [C++] public: int GetHashCode();

2 [VB] Overrides Public Function GetHashCode() As Integer

3 [JScript] public override function GetHashCode() : int;

4
5 *Description*

6 Returns the hash code for this instance.

7 *Return Value:* A 32-bit signed integer hash code.

8 GetTypeCode

9
10 [C#] public TypeCode GetTypeCode();

11 [C++] public: __sealed TypeCode GetTypeCode();

12 [VB] NotOverridable Public Function GetTypeCode() As TypeCode

13 [JScript] public function GetTypeCode() : TypeCode;

14
15 *Description*

16 Returns the **TypeCode** for class **String** .

17 *Return Value:* The enumerated constant, **System.TypeCode.String** .

18 IndexOf

19
20 [C#] public int IndexOf(char value);

21 [C++] public: int IndexOf(__wchar_t value);

22 [VB] Public Function IndexOf(ByVal value As Char) As Integer

23 [JScript] public function IndexOf(value : Char) : int; Reports the index of the first
24 occurrence of a **String** , or one or more characters, within this instance.
25

1
2 *Description*

3 Reports the index of the first occurrence of the specified Unicode character
4 in this instance.

5 *Return Value:* A positive 32-bit signed integer, the index, that is the character
6 position in this instance where *value* was found; otherwise, -1 if *value* was not
7 found.

8 Index numbering starts from zero. A Unicode character to seek.

9 **IndexOf**

10
11 [C#] public int IndexOf(string value);

12 [C++] public: int IndexOf(String* value);

13 [VB] Public Function IndexOf(ByVal value As String) As Integer

14 [JScript] public function IndexOf(value : String) : int;

15
16 *Description*

17 Reports the index of the first occurrence of the specified **String** in this
18 instance.

19 *Return Value:* Value Meaning A positive index position.

20 The search begins at the first character position of this instance and
21 continues until the last character position. The search is case-sensitive and uses the
22 current culture. The **String** to seek.

23 **IndexOf**

24
25 [C#] public int IndexOf(char value, int startIndex);

1 [C++] public: int IndexOf(__wchar_t value, int startIndex);

2 [VB] Public Function IndexOf(ByVal value As Char, ByVal startIndex As
3 Integer) As Integer

4 [JScript] public function IndexOf(value : Char, startIndex : int) : int;

5
6 *Description*

7 Reports the index of the first occurrence of the specified Unicode character
8 in this instance. The search starts at a specified character position.

9 *Return Value:* A positive 32-bit signed integer, the index, indicating the character
10 position in this instance where *value* was found; otherwise, -1 if *value* was not
11 found.

12 Index numbering starts from zero. A Unicode character to seek. The search
13 starting position.

14 IndexOf

15
16 [C#] public int IndexOf(string value, int startIndex);

17 [C++] public: int IndexOf(String* value, int startIndex);

18 [VB] Public Function IndexOf(ByVal value As String, ByVal startIndex As
19 Integer) As Integer

20 [JScript] public function IndexOf(value : String, startIndex : int) : int;

21
22 *Description*

23 Reports the index of the first occurrence of the specified **String** in this
24 instance. The search starts at a specified character position.

25 *Return Value:* Value Meaning A positive index position.

1 The search begins at the *startIndex* character position of this instance and
2 continues until the last character position. The search is case-sensitive and uses the
3 current culture. The **String** to seek. The search starting position.

4 IndexOf

5
6 [C#] public int IndexOf(char value, int startIndex, int count);

7 [C++] public: int IndexOf(__wchar_t value, int startIndex, int count);

8 [VB] Public Function IndexOf(ByVal value As Char, ByVal startIndex As Integer,
9 ByVal count As Integer) As Integer

10 [JScript] public function IndexOf(value : Char, startIndex : int, count : int) : int;

11 12 Description

13 Reports the index of the first occurrence of the specified character in this
14 instance. The search starts at a specified character position and examines a
15 specified number of character positions.

16 *Return Value:* A positive 32-bit signed integer, the index, that is the character
17 position in this instance where *value* was found; otherwise, -1 if *value* was not
18 found.

19 The search begins at *startIndex* and continues until *count* -1. The character
20 at *count* is not included in the search. A Unicode character to seek. The search
21 starting position. The number of character positions to examine.

22 IndexOf

23
24 [C#] public int IndexOf(string value, int startIndex, int count);

25 [C++] public: int IndexOf(String* value, int startIndex, int count);

1 [VB] Public Function IndexOf(ByVal value As String, ByVal startIndex As
2 Integer, ByVal count As Integer) As Integer

3 [JScript] public function IndexOf(value : String, startIndex : int, count : int) : int;

4
5 *Description*

6 Reports the index of the first occurrence of the specified **String** in this
7 instance. The search starts at a specified character position and examines a
8 specified number of character positions.

9 *Return Value:* Value Meaning A positive index position.

10 The search begins at the *startIndex* character position and continues for
11 *count* character positions. The search is case-sensitive and uses the current culture.
12 The **String** to seek. The search starting position. The number of character
13 positions to examine.

14 IndexOfAny

15
16 [C#] public int IndexOfAny(char[] anyOf);

17 [C++] public: int IndexOfAny(__wchar_t anyOf __gc[]);

18 [VB] Public Function IndexOfAny(ByVal anyOf() As Char) As Integer

19 [JScript] public function IndexOfAny(anyOf : Char[]) : int; Reports the index of
20 the first occurrence in this instance of any character in a specified array of
21 Unicode characters.

22
23 *Description*

24 Reports the index of the first occurrence in this instance of any character in
25 a specified array of Unicode characters.

Return Value: The positive integer index of the first occurrence in this instance where any character in *anyOf* was found; otherwise, -1 if no character in *anyOf* was found.

Index numbering starts from zero. A Unicode character array containing one or more characters to seek.

IndexOfAny

[C#] public int IndexOfAny(char[] anyOf, int startIndex);

[C++] public: int IndexOfAny(__wchar_t anyOf __gc[], int startIndex);

[VB] Public Function IndexOfAny(ByVal anyOf() As Char, ByVal startIndex As Integer) As Integer

[JScript] public function IndexOfAny(anyOf : Char[], startIndex : int) : int;

Description

Reports the index of the first occurrence in this instance of any character in a specified array of Unicode characters. The search starts at a specified character position.

Return Value: The positive integer index of the first occurrence in this instance where any character in *anyOf* was found; otherwise, -1 if no character in *anyOf* was found.

Index numbering starts from zero. A Unicode character array containing one or more characters to seek. The search starting position.

IndexOfAny

[C#] public int IndexOfAny(char[] anyOf, int startIndex, int count);

```

1 [C++] public: int IndexOfAny(__wchar_t anyOf __gc[], int startIndex, int count);
2 [VB] Public Function IndexOfAny(ByVal anyOf() As Char, ByVal startIndex As
3 Integer, ByVal count As Integer) As Integer
4 [JScript] public function IndexOfAny(anyOf : Char[], startIndex : int, count : int) :
5 int;
6

```

Description

Reports the index of the first occurrence in this instance of any character in a specified array of Unicode characters. The search starts at a specified character position and examines a specified number of character positions.

Return Value: The positive integer index of the first occurrence in this instance where any character in *anyOf* was found; otherwise, -1 if no character in *anyOf* was found.

The search begins at *startIndex* and continues until *count* - 1. The character at *count* is not included in the search. A Unicode character array containing one or more characters to seek. The search starting position. The number of character positions to examine.

Insert

```

20 [C#] public string Insert(int startIndex, string value);
21 [C++] public: String* Insert(int startIndex, String* value);
22 [VB] Public Function Insert(ByVal startIndex As Integer, ByVal value As String)
23 As String
24 [JScript] public function Insert(startIndex : int, value : String) : String;
25

```

Description

Inserts a specified instance of **String** at a specified index position in this instance.

Return Value: A new **String** equivalent to this instance but with *value* inserted at position *startIndex* .

If *startIndex* is equal to the length of this instance, *value* is appended to the end of this instance. The index position of the insertion. The **String** to insert.

Intern

[C#] public static string Intern(string str);

[C++] public: static String* Intern(String* str);

[VB] Public Shared Function Intern(ByVal str As String) As String

[JScript] public static function Intern(str : String) : String;

Description

Retrieves the system's reference to the specified **String** .

Return Value: The **String** reference to *str* .

The common language runtime automatically maintains a table, called the "intern pool", which contains a single instance of each unique literal string constant declared in a program, as well as any unique instance of **String** you add programmatically. A **String**, or **null**.

IsInterned

[C#] public static string IsInterned(string str);

1 [C++] public: static String* IsInterned(String* str);

2 [VB] Public Shared Function IsInterned(ByVal str As String) As String

3 [JScript] public static function IsInterned(str : String) : String;

4
5 *Description*

6 Retrieves a reference to a specified **String** .

7 *Return Value:* A **String** reference to *str* if it is in the common language runtime
8 "intern pool"; otherwise **null** .

9 The common language runtime automatically maintains a table, called the
10 "intern pool", which contains a single instance of each unique literal string
11 constant declared in a program, as well as any unique instance of **String** you add
12 programmatically. A **String**.

13 Join

14
15 [C#] public static string Join(string separator, string[] value);

16 [C++] public: static String* Join(String* separator, String* value __gc[]);

17 [VB] Public Shared Function Join(ByVal separator As String, ByVal value() As
18 String) As String

19 [JScript] public static function Join(separator : String, value : String[]) : String;

20 Concatenates a specified separator **String** between each element of a specified
21 **String** array, yielding a single concatenated string.

22
23 *Description*

24 Concatenates a specified separator **String** between each element of a
25 specified **String** array, yielding a single concatenated string.

1 *Return Value:* A **String** consisting of the elements of *value* interspersed with the
2 *separator* string.

3 For example if *separator* is ", " and the elements of *value* are "apple",
4 "orange", "grape", and "pear", Join(separator, value) returns "apple, orange, grape,
5 pear". A **System.String**. An array of **Strings**.

6 Join

7
8 [C#] public static string Join(string separator, string[] value, int startIndex, int
9 count);

10 [C++] public: static String* Join(String* separator, String* value __gc[], int
11 startIndex, int count);

12 [VB] Public Shared Function Join(ByVal separator As String, ByVal value() As
13 String, ByVal startIndex As Integer, ByVal count As Integer) As String

14 [JScript] public static function Join(separator : String, value : String[], startIndex :
15 int, count : int) : String;

17 Description

18 Concatenates a specified separator **String** between each element of a
19 specified **String** array, yielding a single concatenated string. Parameters specify
20 the first array element and number of elements to use.

21 *Return Value:* A **String** consisting of the strings in *value* joined by *separator* .

22 For example if *separator* is ", " and the elements of *value* are "apple",
23 "orange", "grape", and "pear", Join(separator, value, 1, 2) returns "orange, grape".
24 A **System.String**. An array of **String**. The first array element in *value* to use. The
25 number of elements of *value* to use.

LastIndexOf

[C#] public int LastIndexOf(char value);

[C++] public: int LastIndexOf(__wchar_t value);

[VB] Public Function LastIndexOf(ByVal value As Char) As Integer

[JScript] public function LastIndexOf(value : Char) : int; Reports the index position of the last occurrence of a specified Unicode character or **String** within this instance.

Description

Reports the index position of the last occurrence of a specified Unicode character within this instance.

Return Value: The index position of *value* if that character is found, or -1 if it is not.

This method begins searching at the last character position of this instance and precedes backwards towards the beginning until either *value* is found or the first character position has been examined. The search is case-sensitive. A Unicode character to seek.

LastIndexOf

[C#] public int LastIndexOf(string value);

[C++] public: int LastIndexOf(String* value);

[VB] Public Function LastIndexOf(ByVal value As String) As Integer

[JScript] public function LastIndexOf(value : String) : int;

1
2 *Description*

3 Reports the index position of the last occurrence of a specified **String**
4 within this instance.

5 *Return Value:* Value Meaning A positive index position.

6 This method begins searching at the last character position of this instance
7 and precedes backwards towards the beginning until either *value* is found or the
8 first character position has been examined. The search is case-sensitive. A **String**
9 to seek.

10 LastIndexOf

11
12 [C#] public int LastIndexOf(char value, int startIndex);

13 [C++] public: int LastIndexOf(__wchar_t value, int startIndex);

14 [VB] Public Function LastIndexOf(ByVal value As Char, ByVal startIndex As
15 Integer) As Integer

16 [JScript] public function LastIndexOf(value : Char, startIndex : int) : int;

17
18 *Description*

19 Reports the index position of the last occurrence of a specified Unicode
20 character within this instance. The search starts at a specified character position.

21 *Return Value:* The index position of *value* if that character is found, or -1 if it is
22 not.

23 This method begins searching at the *startIndex* character position of this
24 instance and precedes backwards towards the beginning until either *value* is found
25

or the first character position has been examined. The search is case-sensitive. A Unicode character to seek. The starting position of a substring within this instance.

LastIndexOf

[C#] public int LastIndexOf(string value, int startIndex);

[C++] public: int LastIndexOf(String* value, int startIndex);

[VB] Public Function LastIndexOf(ByVal value As String, ByVal startIndex As Integer) As Integer

[JScript] public function LastIndexOf(value : String, startIndex : int) : int;

Description

Reports the index position of the last occurrence of a specified **String** within this instance. The search starts at a specified character position.

Return Value: Value Meaning A positive index position.

This method begins searching at the *startIndex* character position of this instance and precedes backwards towards the beginning until either *value* is found or the first character position has been examined. The search is case-sensitive. The **String** to seek. The search starting position.

LastIndexOf

[C#] public int LastIndexOf(char value, int startIndex, int count);

[C++] public: int LastIndexOf(__wchar_t value, int startIndex, int count);

[VB] Public Function LastIndexOf(ByVal value As Char, ByVal startIndex As Integer, ByVal count As Integer) As Integer

[JScript] public function LastIndexOf(value : Char, startIndex : int, count : int) :

1 int;

3 *Description*

4 Reports the index position of the last occurrence of the specified Unicode
5 character in a substring within this instance. The search starts at a specified
6 character position and examines a specified number of character positions.

7 *Return Value:* The index position of *value* if that character is found, or -1 if it is
8 not.

9 This method begins searching at the *startIndex* character position of this
10 instance and precedes backwards towards the beginning until either *value* is found
11 or *count* character positions have been examined. The search is case-sensitive. A
12 Unicode character to seek. The starting position of a substring within this instance.
13 The number of character positions to examine.

14 LastIndexOf

15
16 [C#] public int LastIndexOf(string value, int startIndex, int count);

17 [C++] public: int LastIndexOf(String* value, int startIndex, int count);

18 [VB] Public Function LastIndexOf(ByVal value As String, ByVal startIndex As
19 Integer, ByVal count As Integer) As Integer

20 [JScript] public function LastIndexOf(value : String, startIndex : int, count : int) :

21 int;

22
23 *Description*

24 Reports the index position of the last occurrence of a specified **String**
25 within this instance. The search starts at a specified character position and

examines a specified number of character positions.

Return Value: Value Meaning A positive index position.

This method begins searching at the *startIndex* character position of this instance and precedes backwards towards the beginning until either *value* is found or *count* character positions have been examined. The search is case-sensitive. The **String** to seek. The search starting position. The number of character positions to examine.

LastIndexOfAny

[C#] public int LastIndexOfAny(char[] anyOf);

[C++] public: int LastIndexOfAny(__wchar_t anyOf __gc[]);

[VB] Public Function LastIndexOfAny(ByVal anyOf() As Char) As Integer

[JScript] public function LastIndexOfAny(anyOf : Char[]) : int; Reports the index position of the last occurrence in this instance of one or more characters specified in a Unicode array.

Description

Reports the index position of the last occurrence in this instance of one or more characters specified in a Unicode array.

Return Value: The positive integer index of the last occurrence in this instance where any character in *anyOf* was found; otherwise, -1 if no character in *anyOf* was found.

This method begins searching at the last character position of this instance and precedes backwards towards the beginning until either a character in *anyOf* is

found or the first character position has been examined. The search is case-sensitive. A Unicode character array containing one or more characters to seek.

LastIndexOfAny

[C#] public int LastIndexOfAny(char[] anyOf, int startIndex);

[C++] public: int LastIndexOfAny(__wchar_t anyOf __gc[], int startIndex);

[VB] Public Function LastIndexOfAny(ByVal anyOf() As Char, ByVal startIndex As Integer) As Integer

[JScript] public function LastIndexOfAny(anyOf : Char[], startIndex : int) : int;

Description

Reports the index position of the last occurrence in this instance of one or more characters specified in a Unicode array. The search starts at a specified character position.

Return Value: The positive integer index of the last occurrence in this instance where any character in *anyOf* was found; otherwise, -1 if no character in *anyOf* was found.

This method begins searching at the *startIndex* character position of this instance and precedes backwards towards the beginning until either a character in *anyOf* is found or the first character position has been examined. The search is case-sensitive. A Unicode character array containing one or more characters to seek. The search starting position.

LastIndexOfAny

[C#] public int LastIndexOfAny(char[] anyOf, int startIndex, int count);

```

1 [C++] public: int LastIndexOfAny(__wchar_t anyOf __gc[], int startIndex, int
2 count);
3 [VB] Public Function LastIndexOfAny(ByVal anyOf() As Char, ByVal startIndex
4 As Integer, ByVal count As Integer) As Integer
5 [JScript] public function LastIndexOfAny(anyOf : Char[], startIndex : int, count :
6 int) : int;

```

Description

Reports the index position of the last occurrence in this instance of one or more characters specified in a Unicode array. The search starts at a specified character position and examines a specified number of character positions.

Return Value: The positive integer index of the last occurrence in this instance where any character in *anyOf* was found; otherwise, -1 if no character in *anyOf* was found.

This method begins searching at the *startIndex* character position of this instance and precedes backwards towards the beginning until either a character in *anyOf* is found or *count* character positions have been examined. The search is case-sensitive. A Unicode character array containing one or more characters to seek. The search starting position. The number of character positions to examine.

op_Equality

```

22 [C#] public static bool operator ==(string a, string b);
23 [C++] public: static bool op_Equality(String* a, String* b);
24 [VB] returnValue = String.op_Equality(a, b)
25 [JScript] returnValue = a == b; Determines whether two specified instances of

```

1 **String** or **Object** have the same value.

3 *Description*

4 Determines whether two specified **String** objects have the same value.

5 *Return Value:* **true** if the value of *a* is the same as the value of *b* ; otherwise, **false**.

6 This operator is implemented using the

7 **System.String.Equals(System.Object)** method, which means the comparands are
8 tested for a combination of reference and value equality. The comparison is case-
9 sensitive. A **String** or **null**. A **String** or **null**.

10 op_Inequality

12 [C#] public static bool operator !=(string a, string b);

13 [C++] public: static bool op_Inequality(String* a, String* b);

14 [VB] returnValue = String.op_Inequality(a, b)

15 [JScript] returnValue = a != b; Determines whether two specified instances of

16 **String** or **Object** have different values.

18 *Description*

19 Determines whether two specified **String** objects have different values.

20 *Return Value:* **true** if the value of *a* is different than the value of *b* ; otherwise,
21 **false**.

22 This operator is implemented using the

23 **System.String.Equals(System.Object)** method, which means the comparands are
24 tested for a combination of reference and value equality. The comparison is case-
25 sensitive. A **String** or **null**. A **String** or **null**.

PadLeft

[C#] public string PadLeft(int totalWidth);

[C++] public: String* PadLeft(int totalWidth);

[VB] Public Function PadLeft(ByVal totalWidth As Integer) As String

[JScript] public function PadLeft(totalWidth : int) : String; Right-aligns the characters in this instance, padding on the left with spaces or a specified Unicode character for a specified total length.

Description

Right-aligns the characters in this instance, padding with spaces on the left for a specified total length.

Return Value: A new **String** that is equivalent to this instance, but right-aligned and padded on the left with as many spaces as needed to create a length of *totalWidth* . The number of characters in the resulting string, equal to the number of original characters plus any additional padding characters.

PadLeft

[C#] public string PadLeft(int totalWidth, char paddingChar);

[C++] public: String* PadLeft(int totalWidth, __wchar_t paddingChar);

[VB] Public Function PadLeft(ByVal totalWidth As Integer, ByVal paddingChar As Char) As String

[JScript] public function PadLeft(totalWidth : int, paddingChar : Char) : String;

Description

Right-aligns the characters in this instance, padding on the left with a specified Unicode character for a specified total length.

Return Value: A new **String** that is equivalent to this instance, but right-aligned and padded on the left with as many *paddingChar* characters as needed to create a length of *totalWidth* . The number of characters in the resulting string, equal to the number of original characters plus any additional padding characters. A Unicode padding character.

PadRight

[C#] public string PadRight(int totalWidth);

[C++] public: String* PadRight(int totalWidth);

[VB] Public Function PadRight(ByVal totalWidth As Integer) As String

[JScript] public function PadRight(totalWidth : int) : String; Left-aligns the characters in this string, padding on the right with spaces or a specified Unicode character, for a specified total length.

Description

Left-aligns the characters in this string, padding with spaces on the right, for a specified total length.

Return Value: A new **String** that is equivalent to this instance, but left-aligned and padded on the right with as many spaces as needed to create a length of *totalWidth* . The number of characters in the resulting string, equal to the number of original characters plus any additional padding characters.

PadRight

1
2 [C#] public string PadRight(int totalWidth, char paddingChar);

3 [C++] public: String* PadRight(int totalWidth, __wchar_t paddingChar);

4 [VB] Public Function PadRight(ByVal totalWidth As Integer, ByVal paddingChar
5 As Char) As String

6 [JScript] public function PadRight(totalWidth : int, paddingChar : Char) : String;

7
8 *Description*

9 Left-aligns the characters in this string, padding on the right with a
10 specified Unicode character, for a specified total length.

11 *Return Value:* A new **String** that is equivalent to this instance, but left-aligned and
12 padded on the right with as many *paddingChar* characters as needed to create a
13 length of *totalWidth* . The number of characters in the resulting string, equal to the
14 number of original characters plus any additional padding characters. A Unicode
15 padding character.

16 Remove

17
18 [C#] public string Remove(int startIndex, int count);

19 [C++] public: String* Remove(int startIndex, int count);

20 [VB] Public Function Remove(ByVal startIndex As Integer, ByVal count As
21 Integer) As String

22 [JScript] public function Remove(startIndex : int, count : int) : String;

23
24 *Description*

Deletes a specified number of characters from this instance beginning at a specified position.

Return Value: A new **String** that is equivalent to this instance less *count* number of characters.

For example, the following C# code prints "123456". The position in this instance to begin deleting characters. The number of characters to delete.

Replace

[C#] public string Replace(char oldChar, char newChar);

[C++] public: String* Replace(__wchar_t oldChar, __wchar_t newChar);

[VB] Public Function Replace(ByVal oldChar As Char, ByVal newChar As Char)

As String

[JScript] public function Replace(oldChar : Char, newChar : Char) : String;

Replaces all occurrences of a specified Unicode character or **String** in this instance, with another specified Unicode character or **String**.

Description

Replaces all occurrences of a specified Unicode character in this instance with another specified Unicode character.

Return Value: A **String** equivalent to this instance but with all instances of *oldChar* replaced with *newChar*. A Unicode character to be replaced. A Unicode character to replace all occurrences of *oldChar*.

Replace

[C#] public string Replace(string oldValue, string newValue);

1 [C++] public: String* Replace(String* oldValue, String* newValue);

2 [VB] Public Function Replace(ByVal oldValue As String, ByVal newValue As
3 String) As String

4 [JScript] public function Replace(oldValue : String, newValue : String) : String;

5
6 *Description*

7 Replaces all occurrences of a specified **String** in this instance, with another
8 specified **String** .

9 *Return Value:* A **String** equivalent to this instance but with all instances of
10 *oldValue* replaced with *newValue* . A **String** to be replaced. A **String** to replace
11 all occurrences of *oldValue*.

12 Split

13
14 [C#] public string[] Split(params char[] separator);

15 [C++] public: String* Split(__wchar_t separator __gc[]) __gc[];

16 [VB] Public Function Split(ByVal ParamArray separator() As Char) As String()

17 [JScript] public function Split(separator : Char[]) : String[]; Identifies the
18 substrings in this instance that are delimited by one or more characters specified in
19 an array, then places the substrings into a **String** array.

20
21 *Description*

22 Identifies the substrings in this instance that are delimited by one or more
23 characters specified in an array, then places the substrings into a **String** array.

24 *Return Value:* An array consisting of a single element containing this instance, if
25 this instance contains none of the characters in *separator* .

For example: Input separator Output "42,\n12, 19" new Char[] {'\n','\n'}
 {"42", "", "12", "", "19"} "42..12..19" new Char[] {'.'} {"42", "", "12", "", "19"}
 "Banana" new Char[] {'.'} {"Banana"} "Darb\nSmarba" new Char[] {} {"Darb",
 "Smarba"} "Darb\nSmarba" null {"Darb", "Smarba"} An array of Unicode
 characters that delimit the substrings in this instance, an empty array containing no
 delimiters, or **null**.

Split

[C#] public string[] Split(char[] separator, int count);

[C++] public: String* Split(__wchar_t separator __gc[], int count) __gc[];

[VB] Public Function Split(ByVal separator() As Char, ByVal count As Integer)
 As String()

[JScript] public function Split(separator : Char[], count : int) : String[];

Description

Identifies the substrings in this instance that are delimited by one or more
 characters specified in an array, then places the substrings into a **String** array. A
 parameter specifies the maximum number of array elements to return.

Return Value: An array consisting of a single element containing this instance, if
 this instance contains none of the characters in *separator* .

If there are more than *count* substrings in this instance, the first *count* minus
 1 substrings are returned in the first *count* minus 1 elements of the return value,
 and the remaining characters in this instance are returned in the last element of the
 return value. An array of Unicode characters that delimit the substrings in this

instance, an empty array containing no delimiters, or **null**. The maximum number of array elements to return.

StartsWith

[C#] public bool StartsWith(string value);

[C++] public: bool StartsWith(String* value);

[VB] Public Function StartsWith(ByVal value As String) As Boolean

[JScript] public function StartsWith(value : String) : Boolean;

Description

Determines whether the beginning of this instance matches the specified

String.

Return Value: **true** if *value* matches the beginning of this string or is

System.String.Empty; otherwise **false**.

The StartsWith method makes a comparison at the beginning of the string, determines whether it matches this current instance, and returns a **System.Boolean** representation of their relationship. The specified string must match the prefix or be an empty string (i.e., equals **System.String.Empty**). The comparison is case-sensitive. The **String** to seek.

Substring

[C#] public string Substring(int startIndex);

[C++] public: String* Substring(int startIndex);

[VB] Public Function Substring(ByVal startIndex As Integer) As String

[JScript] public function Substring(startIndex : int) : String; Retrieves a substring

from this instance.

Description

Retrieves a substring from this instance. The substring starts at a specified character position.

Return Value: A **String** equivalent to the substring that begins at *startIndex* in this instance.

The index is zero-based. The starting character position of a substring in this instance.

Substring

[C#] public string Substring(int startIndex, int length);

[C++] public: String* Substring(int startIndex, int length);

[VB] Public Function Substring(ByVal startIndex As Integer, ByVal length As Integer) As String

[JScript] public function Substring(startIndex : int, length : int) : String;

Description

Retrieves a substring from this instance. The substring starts at a specified character position and has a specified length.

Return Value: A **String** equivalent to the substring of length *length* that begins at *startIndex* in this instance.

startIndex is zero-based. The index of the start of the substring. The number of characters in the substring.

IEnumerator.GetEnumerator

```

1
2 [C#] IEnumerator IEnumerable.GetEnumerator();
3 [C++] IEnumerator* IEnumerable::GetEnumerator();
4 [VB] Function GetEnumerator() As IEnumerator Implements
5 IEnumerable.GetEnumerator
6 [JScript] function IEnumerable.GetEnumerator() : IEnumerator;
7     IConvertible.ToBoolean
8
9 [C#] bool IConvertible.ToBoolean(IFormatProvider provider);
10 [C++] bool IConvertible::ToBoolean(IFormatProvider* provider);
11 [VB] Function ToBoolean(ByVal provider As IFormatProvider) As Boolean
12 Implements IConvertible.ToBoolean
13 [JScript] function IConvertible.ToBoolean(provider : IFormatProvider) : Boolean;
14     IConvertible.ToByte
15
16 [C#] byte IConvertible.ToByte(IFormatProvider provider);
17 [C++] unsigned char IConvertible::ToByte(IFormatProvider* provider);
18 [VB] Function ToByte(ByVal provider As IFormatProvider) As Byte Implements
19 IConvertible.ToByte
20 [JScript] function IConvertible.ToByte(provider : IFormatProvider) : Byte;
21     IConvertible.ToChar
22
23 [C#] char IConvertible.ToChar(IFormatProvider provider);
24 [C++] __wchar_t IConvertible::ToChar(IFormatProvider* provider);
25 [VB] Function ToChar(ByVal provider As IFormatProvider) As Char Implements

```

```

1 IConvertible.ToChar
2 [JScript] function IConvertible.ToChar(provider : IFormatProvider) : Char;
3     IConvertible.ToDateTime
4
5 [C#] DateTime IConvertible.ToDateTime(IFormatProvider provider);
6 [C++] DateTime IConvertible::ToDateTime(IFormatProvider* provider);
7 [VB] Function ToDateTime(ByVal provider As IFormatProvider) As DateTime
8 Implements IConvertible.ToDateTime
9 [JScript] function IConvertible.ToDateTime(provider : IFormatProvider) :
10 DateTime;
11     IConvertible.ToDecimal
12
13 [C#] decimal IConvertible.ToDecimal(IFormatProvider provider);
14 [C++] Decimal IConvertible::ToDecimal(IFormatProvider* provider);
15 [VB] Function ToDecimal(ByVal provider As IFormatProvider) As Decimal
16 Implements IConvertible.ToDecimal
17 [JScript] function IConvertible.ToDecimal(provider : IFormatProvider) : Decimal;
18     IConvertible.ToDouble
19
20 [C#] double IConvertible.ToDouble(IFormatProvider provider);
21 [C++] double IConvertible::ToDouble(IFormatProvider* provider);
22 [VB] Function ToDouble(ByVal provider As IFormatProvider) As Double
23 Implements IConvertible.ToDouble
24 [JScript] function IConvertible.ToDouble(provider : IFormatProvider) : double;
25     IConvertible.ToInt16

```



```

1
2 [C#] short IConvertible.ToInt16(IFormatProvider provider);
3 [C++] short IConvertible::ToInt16(IFormatProvider* provider);
4 [VB] Function ToInt16(ByVal provider As IFormatProvider) As Short
5 Implements IConvertible.ToInt16
6 [JScript] function IConvertible.ToInt16(provider : IFormatProvider) : Int16;
7     IConvertible.ToInt32
8
9 [C#] int IConvertible.ToInt32(IFormatProvider provider);
10 [C++] int IConvertible::ToInt32(IFormatProvider* provider);
11 [VB] Function ToInt32(ByVal provider As IFormatProvider) As Integer
12 Implements IConvertible.ToInt32
13 [JScript] function IConvertible.ToInt32(provider : IFormatProvider) : int;
14     IConvertible.ToInt64
15
16 [C#] long IConvertible.ToInt64(IFormatProvider provider);
17 [C++] __int64 IConvertible::ToInt64(IFormatProvider* provider);
18 [VB] Function ToInt64(ByVal provider As IFormatProvider) As Long Implements
19 IConvertible.ToInt64
20 [JScript] function IConvertible.ToInt64(provider : IFormatProvider) : long;
21     IConvertible.ToSByte
22
23 [C#] sbyte IConvertible.ToSByte(IFormatProvider provider);
24 [C++] char IConvertible::ToSByte(IFormatProvider* provider);
25 [VB] Function ToSByte(ByVal provider As IFormatProvider) As SByte

```

1 Implements IConvertible.ToSByte

2 [JScript] function IConvertible.ToSByte(provider : IFormatProvider) : SByte;

3 IConvertible.ToSingle

5 [C#] float IConvertible.ToSingle(IFormatProvider provider);

6 [C++] float IConvertible::ToSingle(IFormatProvider* provider);

7 [VB] Function ToSingle(ByVal provider As IFormatProvider) As Single

8 Implements IConvertible.ToSingle

9 [JScript] function IConvertible.ToSingle(provider : IFormatProvider) : float;

10 IConvertible.ToType

12 [C#] object IConvertible.ToType(Type type, IFormatProvider provider);

13 [C++] Object* IConvertible::ToType(Type* type, IFormatProvider* provider);

14 [VB] Function ToType(ByVal type As Type, ByVal provider As IFormatProvider)

15 As Object Implements IConvertible.ToType

16 [JScript] function IConvertible.ToType(type : Type, provider : IFormatProvider) :

17 Object;

18 IConvertible.ToUInt16

20 [C#] ushort IConvertible.ToUInt16(IFormatProvider provider);

21 [C++] unsigned short IConvertible::ToUInt16(IFormatProvider* provider);

22 [VB] Function ToUInt16(ByVal provider As IFormatProvider) As UInt16

23 Implements IConvertible.ToUInt16

24 [JScript] function IConvertible.ToUInt16(provider : IFormatProvider) : UInt16;

25 IConvertible.ToInt32

```

1
2 [C#] uint IConvertible.ToUInt32(IFormatProvider provider);
3 [C++] unsigned int IConvertible::ToUInt32(IFormatProvider* provider);
4 [VB] Function ToUInt32(ByVal provider As IFormatProvider) As UInt32
5 Implements IConvertible.ToUInt32
6 [JScript] function IConvertible.ToUInt32(provider : IFormatProvider) : UInt32;
7     IConvertible.ToUInt64
8
9 [C#] ulong IConvertible.ToUInt64(IFormatProvider provider);
10 [C++] unsigned __int64 IConvertible::ToUInt64(IFormatProvider* provider);
11 [VB] Function ToUInt64(ByVal provider As IFormatProvider) As UInt64
12 Implements IConvertible.ToUInt64
13 [JScript] function IConvertible.ToUInt64(provider : IFormatProvider) : UInt64;
14     ToCharArray
15
16 [C#] public char[] ToCharArray();
17 [C++] public: __wchar_t ToCharArray() __gc[];
18 [VB] Public Function ToCharArray() As Char()
19 [JScript] public function ToCharArray() : Char[]; Copies the characters in this
20 instance to a Unicode character array.
21
22 Description
23     Copies the characters in this instance to a Unicode character array.
24 Return Value: A Unicode character array whose elements are the individual
25 characters of this instance.

```

ToCharArray

```
[C#] public char[] ToCharArray(int startIndex, int length);  
[C++] public: __wchar_t ToCharArray(int startIndex, int length) __gc[];  
[VB] Public Function ToCharArray(ByVal startIndex As Integer, ByVal length As  
Integer) As Char()  
[JScript] public function ToCharArray(startIndex : int, length : int) : Char[];
```

Description

Copies the characters in a specified substring in this instance to a Unicode character array.

Return Value: A Unicode character array whose elements are the *length* number of characters in this instance starting from character position *startIndex*. The starting position of a substring in this instance. The length of the substring in this instance.

ToLower

```
[C#] public string ToLower();  
[C++] public: String* ToLower();  
[VB] Public Function ToLower() As String  
[JScript] public function ToLower() : String; Returns a copy of this String in  
lowercase.
```

Description

Returns a copy of this **String** in lowercase.

Return Value: A **String** in lowercase.

This method takes into account the current
System.Globalization.CultureInfo information.

ToLower

[C#] public string ToLower(CultureInfo culture);

[C++] public: String* ToLower(CultureInfo* culture);

[VB] Public Function ToLower(ByVal culture As CultureInfo) As String

[JScript] public function ToLower(culture : CultureInfo) : String;

Description

Returns a copy of this **String** in lowercase, taking into account specified culture-specific information.

Return Value: A **String** in lowercase. A **System.Globalization.CultureInfo** object that supplies culture-specific formatting information.

ToString

[C#] public override string ToString();

[C++] public: String* ToString();

[VB] Overrides Public Function ToString() As String

[JScript] public override function ToString() : String; Converts the value of this instance to a **String** .

Description

Returns this instance of **String** ; no actual conversion is performed.

Return Value: This **String** .

ToString

[C#] public string ToString(IFormatProvider provider);
[C++] public: __sealed String* ToString(IFormatProvider* provider);
[VB] NotOverridable Public Function ToString(ByVal provider As
IFormatProvider) As String
[JScript] public function ToString(provider : IFormatProvider) : String;

Description

Returns this instance of **String** ; no actual conversion is performed.

Return Value: This **String** .

provider is reserved, and does not currently participate in this operation.

(Reserved) An **System.IFormatProvider** interface implementation which
supplies culture-specific formatting information.

ToUpper

[C#] public string ToUpper();
[C++] public: String* ToUpper();
[VB] Public Function ToUpper() As String
[JScript] public function ToUpper() : String; Returns a copy of this **String** in
uppercase.

Description

Returns a copy of this **String** in uppercase, using default properties.

Return Value: A new string in uppercase.

This method takes into account the current
System.Globalization.CultureInfo information.

ToUpper

[C#] public string ToUpper(CultureInfo culture);

[C++] public: String* ToUpper(CultureInfo* culture);

[VB] Public Function ToUpper(ByVal culture As CultureInfo) As String

[JScript] public function ToUpper(culture : CultureInfo) : String;

Description

Returns a copy of this **String** in uppercase, taking into account culture-specific information.

Return Value: A **String** in uppercase. A **System.Globalization.CultureInfo** object that supplies culture-specific formatting information.

Trim

[C#] public string Trim();

[C++] public: String* Trim();

[VB] Public Function Trim() As String

[JScript] public function Trim() : String;

Description

Removes all occurrences of white space characters from the beginning and end of this instance.

1 *Return Value:* A new **String** equivalent to this instance after white space
2 characters are removed.

3 This method defines white space characters as hexadecimal 0x9, 0xA, 0xB,
4 0xC, 0xD, 0x20, 0xA0, 0x2000, 0x2001, 0x2002, 0x2003, 0x2004, 0x2005,
5 0x2006, 0x2007, 0x2008, 0x2009, 0x200A, 0x200B, 0x3000, and 0xFEFF.

6 Trim

7
8 [C#] public string Trim(params char[] trimChars);

9 [C++] public: String* Trim(__wchar_t trimChars __gc[]);

10 [VB] Public Function Trim(ByVal ParamArray trimChars() As Char) As String

11 [JScript] public function Trim(trimChars : Char[]) : String; Removes all

12 occurrences of a set of specified characters from the beginning and end of this
13 instance.

14
15 *Description*

16 Removes all occurrences of a set of characters specified in a Unicode
17 character array from the beginning and end of this instance.

18 *Return Value:* The **String** that remains after all occurrences of the characters in
19 *trimChars* are removed. If *trimChars* is **null**, white space characters are removed
20 instead.

21 This method defines white space characters as hexadecimal 0x9, 0xA, 0xB,
22 0xC, 0xD, 0x20, 0xA0, 0x2000, 0x2001, 0x2002, 0x2003, 0x2004, 0x2005,
23 0x2006, 0x2007, 0x2008, 0x2009, 0x200A, 0x200B, 0x3000, and 0xFEFF. An
24 array of Unicode characters to be removed or **null**.

25 TrimEnd

1
2 [C#] public string TrimEnd(params char[] trimChars);

3 [C++] public: String* TrimEnd(__wchar_t trimChars __gc[]);

4 [VB] Public Function TrimEnd(ByVal ParamArray trimChars() As Char) As
5 String

6 [JScript] public function TrimEnd(trimChars : Char[]) : String;

7
8 *Description*

9 Removes all occurrences of a set of characters specified in a Unicode
10 character array from the end of this instance.

11 *Return Value:* The **String** that remains after all occurrences of the characters in
12 *trimChars* are removed. If *trimChars* is **null**, white space characters are removed
13 instead.

14 This method defines white space characters as hexadecimal 0x9, 0xA, 0xB,
15 0xC, 0xD, 0x20, 0xA0, 0x2000, 0x2001, 0x2002, 0x2003, 0x2004, 0x2005,
16 0x2006, 0x2007, 0x2008, 0x2009, 0x200A, 0x200B, 0x3000, and 0xFEFF. An
17 array of Unicode characters to be removed or **null**.

18 TrimStart

19
20 [C#] public string TrimStart(params char[] trimChars);

21 [C++] public: String* TrimStart(__wchar_t trimChars __gc[]);

22 [VB] Public Function TrimStart(ByVal ParamArray trimChars() As Char) As
23 String

24 [JScript] public function TrimStart(trimChars : Char[]) : String;

1
2 *Description*

3 Removes all occurrences of a set of characters specified in a Unicode
4 character array from the beginning of this instance.

5 *Return Value:* The **String** that remains after all occurrences of characters in
6 *trimChars* are removed. If *trimChars* is **null** , white space characters are removed
7 instead.

8 This method defines white space characters as hexadecimal 0x9, 0xA, 0xB,
9 0xC, 0xD, 0x20, 0xA0, 0x2000, 0x2001, 0x2002, 0x2003, 0x2004, 0x2005,
10 0x2006, 0x2007, 0x2008, 0x2009, 0x200A, 0x200B, 0x3000, and 0xFEFF. An
11 array of Unicode characters to be removed or **null**.

12 SystemException class (System)

13 TrimStart
14
15

16 *Description*

17 Defines the base class for predefined exceptions in the **System** namespace.

18 **System.SystemException** is thrown by the common language runtime
19 when errors occur that are nonfatal and recoverable by user programs. These
20 errors result from failed runtime check (such as an array out-of-bound error), and
21 can occur during the execution of any method.

22 SystemException

23 *Example Syntax:*

24 TrimStart
25

[C#] public SystemException();
 [C++] public: SystemException();
 [VB] Public Sub New()
 [JScript] public function SystemException(); Initializes a new instance of the **System.SystemException** class.

Description

Initializes a new instance of the **System.SystemException** class with default properties.

The following table shows the initial property values for an instance of **System.SystemException**.

Property	Initial Value
SystemException	

Example Syntax:

TrimStart

[C#] public SystemException(string message);
 [C++] public: SystemException(String* message);
 [VB] Public Sub New(ByVal message As String)
 [JScript] public function SystemException(message : String);

Description

Initializes a new instance of the **System.SystemException** class with a specified error message.

The following table shows the initial property values for an instance of **System.SystemException**. The error message that explains the reason for the exception.

SystemException

Example Syntax:

TrimStart

[C#] protected SystemException(SerializationInfo info, StreamingContext context);

[C++] protected: SystemException(SerializationInfo* info, StreamingContext context);

[VB] Protected Sub New(ByVal info As SerializationInfo, ByVal context As StreamingContext)

[JScript] protected function SystemException(info : SerializationInfo, context : StreamingContext);

Description

Initializes a new instance of the **System.SystemException** class with serialized data.

This constructor is called during deserialization to reconstitute the exception object transmitted over a stream. For more information, see . The object that holds the serialized object data. The contextual information about the source or destination.

SystemException

Example Syntax:

TrimStart

```
[C#] public SystemException(string message, Exception innerException);  
[C++] public: SystemException(String* message, Exception* innerException);  
[VB] Public Sub New(ByVal message As String, ByVal innerException As  
Exception)  
[JScript] public function SystemException(message : String, innerException :  
Exception);
```

Description

Initializes a new instance of the **System.SystemException** class with a specified error message and a reference to the inner exception that is the root cause of this exception.

When an **Exception** *X* is thrown as a direct result of a previous exception *Y*, the **System.Exception.InnerException** property of *X* should contain a reference to *Y*. The **InnerException** property returns the same value as was passed into the constructor, or **null** if the inner exception value was not supplied to the constructor. The error message that explains the reason for the exception. An instance of **System.Exception** that is the cause of the current **Exception**. If *innerException* is non-null, then the current **Exception** is raised in a catch block handling *innerException*.

HelpLink

HResult

InnerException

Message

Source

StackTrace

TargetSite

ThreadStaticAttribute class (System)

ToString

Description

Indicates that the value of a static field is unique for each thread.

A static field marked with **System.ThreadStaticAttribute** is not shared between threads. Each executing thread has a separate instance of the static field, and independently set and get values for that field. If the static field is accessed on a different thread, it will contain a different value.

ThreadStaticAttribute

Example Syntax:

ToString

```
[C#] public ThreadStaticAttribute();
```

```
[C++] public: ThreadStaticAttribute();
```

```
[VB] Public Sub New()
```

```
[JScript] public function ThreadStaticAttribute();
```

Description

Initializes a new instance of the **System.ThreadStaticAttribute** class.

TypeId

1 TimeSpan structure (System)

2 ToString

3
4
5 *Description*

6 Represents a time interval.

7 The value of an instance of **TimeSpan** represents a period of time. That
8 value is the number of "ticks" contained in the instance. A tick is the smallest unit
9 of time that can be specified, and is equal to 100 nanoseconds. Both the
10 specification of a number of ticks and the value of a **TimeSpan** can be positive or
11 negative.

12 ToString

13
14 [C#] public static readonly TimeSpan MaxValue;

15 [C++] public: static TimeSpan MaxValue;

16 [VB] Public Shared ReadOnly MaxValue As TimeSpan

17 [JScript] public static var MaxValue : TimeSpan;

18
19 *Description*

20 A constant whose value is the maximum **TimeSpan** value.

21 The value of this constant is equivalent to **System.Int64.MaxValue** ticks.

22 The string representation of this value is positive 10675199.02:48:05.4775807.

23 ToString

24
25 [C#] public static readonly TimeSpan MinValue;

1 [C++] public: static TimeSpan MinValue;

2 [VB] Public Shared ReadOnly MinValue As TimeSpan

3 [JScript] public static var MinValue : TimeSpan;

4
5 *Description*

6 A constant whose value is the minimum **TimeSpan** value.

7 The value of this constant is equivalent to **System.Int64.MinValue** ticks.

8 The string representation of this value is negative 10675199.02:48:05.4775808.

9 ToString

10
11 [C#] public const long TicksPerDay;

12 [C++] public: const __int64 TicksPerDay;

13 [VB] Public Const TicksPerDay As Long

14 [JScript] public var TicksPerDay : long;

15
16 *Description*

17 A constant whose value is the number of ticks equivalent to 1 day.

18 The value of this constant is 864 billion; that is, 864000000000.

19 ToString

20
21 [C#] public const long TicksPerHour;

22 [C++] public: const __int64 TicksPerHour;

23 [VB] Public Const TicksPerHour As Long

24 [JScript] public var TicksPerHour : long;

1
2 *Description*

3 A constant whose value is the number of ticks equivalent to 1 hour.

4 The value of this constant is 36 billion; that is, 36000000000.

5 ToString

6
7 [C#] public const long TicksPerMillisecond;

8 [C++] public: const __int64 TicksPerMillisecond;

9 [VB] Public Const TicksPerMillisecond As Long

10 [JScript] public var TicksPerMillisecond : long;

11
12 *Description*

13 A constant whose value is the number of ticks equivalent to 1 millisecond.

14 The value of this constant is 10 thousand; that is, 10000.

15 ToString

16
17 [C#] public const long TicksPerMinute;

18 [C++] public: const __int64 TicksPerMinute;

19 [VB] Public Const TicksPerMinute As Long

20 [JScript] public var TicksPerMinute : long;

21
22 *Description*

23 A constant whose value is the number of ticks equivalent to 1 minute.

24 The value of this constant is 600 million; that is, 600000000.

25 ToString

```

1 [C#] public const long TicksPerSecond;
2
3 [C++] public: const __int64 TicksPerSecond;
4
5 [VB] Public Const TicksPerSecond As Long
6
7 [JScript] public var TicksPerSecond : long;
8

```

Description

A constant whose value is the number of ticks equivalent to 1 second.

The value of this constant is 10 million; that is, 10000000.

ToString

```

11
12 [C#] public static readonly TimeSpan Zero;
13
14 [C++] public: static TimeSpan Zero;
15
16 [VB] Public Shared ReadOnly Zero As TimeSpan
17
18 [JScript] public static var Zero : TimeSpan;
19

```

Description

A constant whose value is the zero **TimeSpan** value.

This constant provides a convenient source for zero in time calculations.

TimeSpan

Example Syntax:

ToString

```

23
24 [C#] public TimeSpan(long ticks);
25
26 [C++] public: TimeSpan(__int64 ticks);

```

1 [VB] Public Sub New(ByVal ticks As Long)

2 [JScript] public function TimeSpan(ticks : long); Initializes a new instance of the
3 **TimeSpan** class.

4
5 *Description*

6 Initializes a new instance of the **TimeSpan** class to the specified number of
7 ticks. A time period in the form of ticks.

8 TimeSpan

9 *Example Syntax:*

10 ToString

11
12 [C#] public TimeSpan(int hours, int minutes, int seconds);

13 [C++] public: TimeSpan(int hours, int minutes, int seconds);

14 [VB] Public Sub New(ByVal hours As Integer, ByVal minutes As Integer, ByVal
15 seconds As Integer)

16 [JScript] public function TimeSpan(hours : int, minutes : int, seconds : int);

17
18 *Description*

19 Initializes a new instance of the **TimeSpan** class to a specified number of
20 hours, minutes, and seconds.

21 The specified *hours* , *minutes* , and *seconds* are converted to ticks, and that
22 value initializes this instance. Number of hours. Number of minutes. Number of
23 seconds.

24 TimeSpan

25 *Example Syntax:*

ToString

```
[C#] public TimeSpan(int days, int hours, int minutes, int seconds);  
[C++] public: TimeSpan(int days, int hours, int minutes, int seconds);  
[VB] Public Sub New(ByVal days As Integer, ByVal hours As Integer, ByVal  
minutes As Integer, ByVal seconds As Integer)  
[JScript] public function TimeSpan(days : int, hours : int, minutes : int, seconds :  
int);
```

Description

Initializes a new instance of the **TimeSpan** class to a specified number of days, hours, minutes, and seconds.

The specified *days* , *hours* , *minutes* , and *seconds* are converted to ticks, and that value initializes this instance. Number of days. Number of hours. Number of minutes. Number of seconds.

TimeSpan

Example Syntax:

ToString

```
[C#] public TimeSpan(int days, int hours, int minutes, int seconds, int  
milliseconds);  
[C++] public: TimeSpan(int days, int hours, int minutes, int seconds, int  
milliseconds);  
[VB] Public Sub New(ByVal days As Integer, ByVal hours As Integer, ByVal  
minutes As Integer, ByVal seconds As Integer, ByVal milliseconds As Integer)
```

1 [JScript] public function TimeSpan(days : int, hours : int, minutes : int, seconds :
2 int, milliseconds : int);
3

4 *Description*

5 Initializes a new instance of the **TimeSpan** class to a specified number of
6 days, hours, minutes, seconds, and milliseconds.

7 The specified *days* , *hours* , *minutes* , *seconds* , and *milliseconds* are
8 converted to ticks, and that value initializes this instance. Number of days.

9 Number of hours. Number of minutes. Number of seconds. Number of
10 milliseconds.

11 Days

12 ToString
13

14 [C#] public int Days {get;}

15 [C++] public: __property int get_Days();

16 [VB] Public ReadOnly Property Days As Integer

17 [JScript] public function get Days() : int;
18

19 *Description*

20 Gets the number of whole days represented by this instance.

21 **DateTime** values can be represented as expressions of the form

22 "d.hh:mm:ss.ff" where the "d" component is days, "hh" is hours, "mm" is minutes,
23 "ss" is seconds, and "ff" is fractions of a second. The value of this property is the
24 days component.
25

 Hours

ToString

[C#] public int Hours {get;}

[C++] public: __property int get_Hours();

[VB] Public ReadOnly Property Hours As Integer

[JScript] public function get Hours() : int;

Description

Gets the number of whole hours represented by this instance.

DateTime values can be represented as expressions of the form

"d.hh:mm:ss.ff" where the "d" component is days, "hh" is hours, "mm" is minutes,

"ss" is seconds, and "ff" is fractions of a second. The value of this property is the

hours component.

Milliseconds

ToString

[C#] public int Milliseconds {get;}

[C++] public: __property int get_Milliseconds();

[VB] Public ReadOnly Property Milliseconds As Integer

[JScript] public function get Milliseconds() : int;

Description

Gets the number of whole milliseconds represented by this instance.

DateTime values can be represented as expressions of the form

"d.hh:mm:ss.ff" where the "d" component is days, "hh" is hours, "mm" is minutes,

"ss" is seconds, and "ff" is fractions of a second. The value of this property is the fractions of a second component expressed in milliseconds.

Minutes

ToString

```
[C#] public int Minutes {get;}
```

```
[C++] public: __property int get_Minutes();
```

```
[VB] Public ReadOnly Property Minutes As Integer
```

```
[JScript] public function get Minutes() : int;
```

Description

Gets the number of whole minutes represented by this instance.

DateTime values can be represented as expressions of the form

"d.hh:mm:ss.ff" where the "d" component is days, "hh" is hours, "mm" is minutes, "ss" is seconds, and "ff" is fractions of a second. The value of this property is the minutes component.

Seconds

ToString

```
[C#] public int Seconds {get;}
```

```
[C++] public: __property int get_Seconds();
```

```
[VB] Public ReadOnly Property Seconds As Integer
```

```
[JScript] public function get Seconds() : int;
```

Description

1 Gets the number of whole seconds represented by this instance.

2 **DateTime** values can be represented as expressions of the form

3 "d.hh:mm:ss.ff" where the "d" component is days, "hh" is hours, "mm" is minutes,
4 "ss" is seconds, and "ff" is fractions of a second. The value of this property is the
5 seconds component.

6 Ticks

7 ToString

8
9 [C#] public long Ticks {get;}

10 [C++] public: __property __int64 get_Ticks();

11 [VB] Public ReadOnly Property Ticks As Long

12 [JScript] public function get Ticks() : long;

13
14 *Description*

15 Gets the value of this instance in ticks.

16 The smallest unit of time is the "tick," which is equal to 100-nanoseconds.

17 A tick can be negative or positive.

18 TotalDays

19 ToString

20
21 [C#] public double TotalDays {get;}

22 [C++] public: __property double get_TotalDays();

23 [VB] Public ReadOnly Property TotalDays As Double

24 [JScript] public function get TotalDays() : double;

Description

Gets the value of this instance expressed in whole and fractional days.

TotalHours

ToString

[C#] public double TotalHours {get;}

[C++] public: __property double get_TotalHours();

[VB] Public ReadOnly Property TotalHours As Double

[JScript] public function get TotalHours() : double;

Description

Gets the value of this instance expressed in whole and fractional hours.

TotalMilliseconds

ToString

[C#] public double TotalMilliseconds {get;}

[C++] public: __property double get_TotalMilliseconds();

[VB] Public ReadOnly Property TotalMilliseconds As Double

[JScript] public function get TotalMilliseconds() : double;

Description

Gets the value of this instance expressed in whole and fractional
milliseconds.

TotalMinutes

ToString

[C#] public double TotalMinutes {get;}

[C++] public: __property double get_TotalMinutes();

[VB] Public ReadOnly Property TotalMinutes As Double

[JScript] public function get TotalMinutes() : double;

Description

Gets the value of this instance expressed in whole and fractional minutes.

TotalSeconds

ToString

[C#] public double TotalSeconds {get;}

[C++] public: __property double get_TotalSeconds();

[VB] Public ReadOnly Property TotalSeconds As Double

[JScript] public function get TotalSeconds() : double;

Description

Gets the value of this instance expressed in whole and fractional seconds.

Add

[C#] public TimeSpan Add(TimeSpan ts);

[C++] public: TimeSpan Add(TimeSpan ts);

[VB] Public Function Add(ByVal ts As TimeSpan) As TimeSpan

[JScript] public function Add(ts : TimeSpan) : TimeSpan;

Description

Adds the specified **TimeSpan** to this instance.

Return Value: A **TimeSpan** that represents the value of this instance plus the value of *ts* .

The result must be between **System.TimeSpan.MinValue** and **System.TimeSpan.MaxValue** , otherwise an exception is thrown. A **TimeSpan** instance.

Compare

```
[C#] public static int Compare(TimeSpan t1, TimeSpan t2);
```

```
[C++] public: static int Compare(TimeSpan t1, TimeSpan t2);
```

```
[VB] Public Shared Function Compare(ByVal t1 As TimeSpan, ByVal t2 As  
TimeSpan) As Integer
```

```
[JScript] public static function Compare(t1 : TimeSpan, t2 : TimeSpan) : int;
```

Description

Compares two **TimeSpan** values and returns an integer that indicates their relationship.

Return Value: Value Condition -1 *t1* is less than *t2* 0 *t1* is equal to *t2* 1 *t1* is greater than *t2* The first **TimeSpan** instance. The second **TimeSpan** instance.

CompareTo

```
[C#] public int CompareTo(object value);
```

```
[C++] public: __sealed int CompareTo(Object* value);
```

1 [VB] NotOverridable Public Function CompareTo(ByVal value As Object) As

2 Integer

3 [JScript] public function CompareTo(value : Object) : int;

4
5 *Description*

6 Compares this instance to a specified object and returns an indication of
7 their relative values.

8 *Return Value:* Value Condition -1 The value of this instance is less than the value
9 of *value* .

10 Any instance of **TimeSpan** , regardless of its value, is considered greater
11 than **null** . An object to compare, or **null**.

12 *Duration*

13
14 [C#] public TimeSpan Duration();

15 [C++] public: TimeSpan Duration();

16 [VB] Public Function Duration() As TimeSpan

17 [JScript] public function Duration() : TimeSpan;

18
19 *Description*

20 Returns a **TimeSpan** whose value is the absolute value of this instance.

21 *Return Value:* A **TimeSpan** whose value is the value of this instance and
22 converted if necessary to a positive number.

23 The value of a **TimeSpan** , which is the number of ticks it contains, can be
24 positive or negative.

25 *Equals*

1
2 [C#] public override bool Equals(object value);

3 [C++] public: bool Equals(Object* value);

4 [VB] Overrides Public Function Equals(ByVal value As Object) As Boolean

5 [JScript] public override function Equals(value : Object) : Boolean; Returns a
6 value indicating whether two instances of **TimeSpan** are equal.

7
8 *Description*

9 Returns a value indicating whether this instance is equal to a specified
10 object.

11 *Return Value:* **true** if *value* is a **TimeSpan** that represents the same time as this
12 instance; otherwise, **false** . An object to compare with this instance.

13 Equals

14
15 [C#] public static new bool Equals(TimeSpan t1, TimeSpan t2);

16 [C++] public: static bool Equals(TimeSpan t1, TimeSpan t2);

17 [VB] Shadows Public Shared Function Equals(ByVal t1 As TimeSpan, ByVal t2
18 As TimeSpan) As Boolean

19 [JScript] public static hide function Equals(t1 : TimeSpan, t2 : TimeSpan) :
20 Boolean;

21
22 *Description*

23 Returns a value indicating whether two specified instances of **TimeSpan**
24 are equal.

Return Value: **true** if the values of *t1* and *t2* are equal; otherwise, **false** . An instance of **TimeSpan**. An instance of **TimeSpan**.

FromDays

[C#] public static TimeSpan FromDays(double value);

[C++] public: static TimeSpan FromDays(double value);

[VB] Public Shared Function FromDays(ByVal value As Double) As TimeSpan

[JScript] public static function FromDays(value : double) : TimeSpan;

Description

Returns a **TimeSpan** that represents a specified number of days, where the specification is accurate to the nearest millisecond.

Return Value: A **TimeSpan** that represents *value* .

If *value* is **System.Double.PositiveInfinity** , **System.TimeSpan.MaxValue** is returned. If *value* is **System.Double.NegativeInfinity** or **System.Double.NaN** , **System.TimeSpan.MinValue** is returned. A number of days, accurate to the nearest millisecond.

FromHours

[C#] public static TimeSpan FromHours(double value);

[C++] public: static TimeSpan FromHours(double value);

[VB] Public Shared Function FromHours(ByVal value As Double) As TimeSpan

[JScript] public static function FromHours(value : double) : TimeSpan;

1
2 *Description*

3 Returns a **TimeSpan** that represents a specified number of hours, where the
4 specification is accurate to the nearest millisecond.

5 *Return Value:* A **TimeSpan** that represents *value* .

6 If *value* is **System.Double.PositiveInfinity** ,
7 **System.TimeSpan.MaxValue** is returned. If *value* is
8 **System.Double.NegativeInfinity** or **System.Double.NaN** ,
9 **System.TimeSpan.MinValue** is returned. A number of hours accurate to the
10 nearest millisecond.

11 FromMilliseconds

12
13 [C#] public static TimeSpan FromMilliseconds(double value);

14 [C++] public: static TimeSpan FromMilliseconds(double value);

15 [VB] Public Shared Function FromMilliseconds(ByVal value As Double) As
16 TimeSpan

17 [JScript] public static function FromMilliseconds(value : double) : TimeSpan;

18
19 *Description*

20 Returns a **TimeSpan** that represents a specified number of milliseconds.

21 *Return Value:* A **TimeSpan** that represents *value* .

22 If *value* is **System.Double.PositiveInfinity** ,
23 **System.TimeSpan.MaxValue** is returned. If *value* is
24 **System.Double.NegativeInfinity** or **System.Double.NaN** ,
25 **System.TimeSpan.MinValue** is returned. A number of milliseconds.

FromMinutes

```
[C#] public static TimeSpan FromMinutes(double value);  
[C++] public: static TimeSpan FromMinutes(double value);  
[VB] Public Shared Function FromMinutes(ByVal value As Double) As  
TimeSpan  
[JScript] public static function FromMinutes(value : double) : TimeSpan;
```

Description

Returns a **TimeSpan** that represents a specified number of minutes, where the specification is accurate to the nearest millisecond.

Return Value: A **TimeSpan** that represents *value* .

If *value* is **System.Double.PositiveInfinity** ,
System.TimeSpan.MaxValue is returned. If *value* is
System.Double.NegativeInfinity or **System.Double.NaN** ,
System.TimeSpan.MinValue is returned. A number of minutes, accurate to the
nearest millisecond.

FromSeconds

```
[C#] public static TimeSpan FromSeconds(double value);  
[C++] public: static TimeSpan FromSeconds(double value);  
[VB] Public Shared Function FromSeconds(ByVal value As Double) As  
TimeSpan  
[JScript] public static function FromSeconds(value : double) : TimeSpan;
```


Description

Returns a **TimeSpan** that represents a specified number of seconds, where the specification is accurate to the nearest millisecond.

Return Value: A **TimeSpan** that represents *value* .

If *value* is **System.Double.PositiveInfinity** , **System.TimeSpan.MaxValue** is returned. If *value* is **System.Double.NegativeInfinity** or **System.Double.NaN** , **System.TimeSpan.MinValue** is returned. A number of seconds, accurate to the nearest millisecond.

FromTicks

[C#] public static TimeSpan FromTicks(long value);

[C++] public: static TimeSpan FromTicks(__int64 value);

[VB] Public Shared Function FromTicks(ByVal value As Long) As TimeSpan

[JScript] public static function FromTicks(value : long) : TimeSpan;

Description

Returns a **TimeSpan** that represents a specified time, where the specification is in units of ticks.

Return Value: A **TimeSpan** with a value of *value* .

This is a convenience method with the same behavior as the **System.TimeSpan.#ctor** constructor. A number of ticks that represent a time.

GetHashCode

1
2 [C#] public override int GetHashCode();

3 [C++] public: int GetHashCode();

4 [VB] Overrides Public Function GetHashCode() As Integer

5 [JScript] public override function GetHashCode() : int;

6
7 *Description*

8 Returns a hash code for this instance.

9 *Return Value:* A 32-bit signed integer hash code.

10 Two **TimeSpan** objects might have the same hash code even though they
11 represent different time values.

12 **Negate**

13
14 [C#] public TimeSpan Negate();

15 [C++] public: TimeSpan Negate();

16 [VB] Public Function Negate() As TimeSpan

17 [JScript] public function Negate() : TimeSpan;

18
19 *Description*

20 Returns a **TimeSpan** whose value is the negated value of this instance.

21 *Return Value:* The same numeric value as this instance, but with the opposite sign.

22 **op_Addition**

23
24 [C#] public static TimeSpan operator +(TimeSpan t1, TimeSpan t2);

25 [C++] public: static TimeSpan op_Addition(TimeSpan t1, TimeSpan t2);

1 [VB] returnValue = TimeSpan.op_Addition(t1, t2)

2 [JScript] returnValue = t1 + t2;

3

4 *Description*

5 Adds two specified **TimeSpan** instances.

6 *Return Value:* A **TimeSpan** whose value is the sum of the values of *t1* and *t2* . A

7 **TimeSpan A TimeSpan**

8 op_Equality

9

10 [C#] public static bool operator ==(TimeSpan t1, TimeSpan t2);

11 [C++] public: static bool op_Equality(TimeSpan t1, TimeSpan t2);

12 [VB] returnValue = TimeSpan.op_Equality(t1, t2)

13 [JScript] returnValue = t1 == t2;

14

15 *Description*

16 Indicates whether two **TimeSpan** instances are equal.

17 *Return Value:* **true** if the values of *t1* and *t2* are equal; otherwise, **false** . A

18 **TimeSpan A TimeSpan**

19 op_GreaterThan

20

21 [C#] public static bool operator >(TimeSpan t1, TimeSpan t2);

22 [C++] public: static bool op_GreaterThan(TimeSpan t1, TimeSpan t2);

23 [VB] returnValue = TimeSpan.op_GreaterThan(t1, t2)

24 [JScript] returnValue = t1 > t2;

25

1
2 *Description*

3 Indicates whether a specified **TimeSpan** is greater than another specified
4 **TimeSpan** .

5 *Return Value:* **true** if the value of *t1* is greater than the value of *t2* ; otherwise,
6 **false** . A **TimeSpan** A **TimeSpan**

7 op_GreaterThanOrEqual

8
9 [C#] public static bool operator >=(TimeSpan t1, TimeSpan t2);

10 [C++] public: static bool op_GreaterThanOrEqual(TimeSpan t1, TimeSpan t2);

11 [VB] returnValue = TimeSpan.op_GreaterThanOrEqual(t1, t2)

12 [JScript] returnValue = t1 >= t2;

13
14 *Description*

15 Indicates whether a specified **TimeSpan** is greater than or equal to another
16 specified **TimeSpan** .

17 *Return Value:* **true** if the value of *t1* is greater than or equal to the value of *t2* ;
18 otherwise, **false** . A **TimeSpan** A **TimeSpan**

19 op_Inequality

20
21 [C#] public static bool operator !=(TimeSpan t1, TimeSpan t2);

22 [C++] public: static bool op_Inequality(TimeSpan t1, TimeSpan t2);

23 [VB] returnValue = TimeSpan.op_Inequality(t1, t2)

24 [JScript] returnValue = t1 != t2;

1
2 *Description*

3 Indicates whether two **TimeSpan** instances are not equal.

4 *Return Value:* **true** if the values of *t1* and *t2* are not equal; otherwise, **false** . A

5 **TimeSpan A TimeSpan**

6 op_LessThan

7
8 [C#] public static bool operator

9 [C++] public: static bool op_LessThan(TimeSpan t1, TimeSpan t2);

10 [VB] returnValue = TimeSpan.op_LessThan(t1, t2)

11 [JScript] returnValue = t1 < t2;

12
13 *Description*

14 Indicates whether a specified **TimeSpan** is less than another specified

15 **TimeSpan** .

16 *Return Value:* **true** if the value of *t1* is less than the value of *t2* ; otherwise, **false** .

17 **A TimeSpan A TimeSpan**

18 op_LessThanOrEqual

19
20 [C#] public static bool operator <=(TimeSpan t1, TimeSpan t2);

21 [C++] public: static bool op_LessThanOrEqual(TimeSpan t1, TimeSpan t2);

22 [VB] returnValue = TimeSpan.op_LessThanOrEqual(t1, t2)

23 [JScript] returnValue = t1 <= t2;

24
25 *Description*

Indicates whether a specified **TimeSpan** is less than or equal to another specified **TimeSpan** .

Return Value: **true** if the value of *t1* is less than or equal to the value of *t2* ; otherwise, **false** . A **TimeSpan** A **TimeSpan**

op_Subtraction

[C#] public static TimeSpan operator -(TimeSpan t1, TimeSpan t2);

[C++] public: static TimeSpan op_Subtraction(TimeSpan t1, TimeSpan t2);

[VB] returnValue = TimeSpan.op_Subtraction(t1, t2)

[JScript] returnValue = t1 - t2;

Description

Subtracts a specified **TimeSpan** from another specified **TimeSpan** .

Return Value: A **TimeSpan** whose value is the result of the value of *t1* minus the value of *t2* . A **TimeSpan** A **TimeSpan**

op_UnaryNegation

[C#] public static TimeSpan operator -(TimeSpan t);

[C++] public: static TimeSpan op_UnaryNegation(TimeSpan t);

[VB] returnValue = TimeSpan.op_UnaryNegation(t)

[JScript] returnValue = -t;

Description

Returns a **TimeSpan** whose value is the negated value of the specified instance.

Return Value: A **TimeSpan** with the same numeric value as this instance, but the opposite sign. A **TimeSpan**

op_UnaryPlus

[C#] public static TimeSpan operator +(TimeSpan t);

[C++] public: static TimeSpan op_UnaryPlus(TimeSpan t);

[VB] returnValue = TimeSpan.op_UnaryPlus(t)

[JScript] returnValue = +t;

Description

Returns the specified instance of **TimeSpan**.

Return Value: Returns *t*. A **TimeSpan**

Parse

[C#] public static TimeSpan Parse(string s);

[C++] public: static TimeSpan Parse(String* s);

[VB] Public Shared Function Parse(ByVal s As String) As TimeSpan

[JScript] public static function Parse(s : String) : TimeSpan;

Description

Constructs a **TimeSpan** from a time indicated by a specified **String**.

Return Value: A **TimeSpan** that corresponds to *s*.

s contains a specification of the form: [ws][-][d.]hh:mm:ss[.ff][ws] Items in square brackets ('[' and ']') are optional, colons and periods (':' and '.') are literal characters, and other items are as follows. A **String**.

Subtract

```
[C#] public TimeSpan Subtract(TimeSpan ts);  
[C++] public: TimeSpan Subtract(TimeSpan ts);  
[VB] Public Function Subtract(ByVal ts As TimeSpan) As TimeSpan  
[JScript] public function Subtract(ts : TimeSpan) : TimeSpan;
```

Description

Subtracts the specified **TimeSpan** object from this instance.

Return Value: A **TimeSpan** whose value is the result of the value of this instance minus the value of *ts* .

The result must be between **System.TimeSpan.MinValue** and **System.TimeSpan.MaxValue** , otherwise an exception is thrown. A **TimeSpan** instance.

ToString

```
[C#] public override string ToString();  
[C++] public: String* ToString();  
[VB] Overrides Public Function ToString() As String  
[JScript] public override function ToString() : String; Returns the String  
representation of the value of this instance.
```

Description

Returns the **String** representation of the value of this instance.

Return Value: A **System.String** that represents the value of this instance. The

format of the return value is of the form: [-][d.]hh:mm:ss[.ff] Items in square brackets ('[' and ']') are optional, colons and periods (':' and '.') are literal characters, and other items are as follows.

The return value of this method can be consumed by

System.TimeSpan.Parse(System.String) .

TimeZone class (System)

ToString

Description

Represents a time zone.

A time zone is a geographical region in which the same standard time is used.

TimeZone

Example Syntax:

ToString

[C#] protected TimeZone();

[C++] protected: TimeZone();

[VB] Protected Sub New()

[JScript] protected function TimeZone();

Description

Initializes a new instance of the **System.TimeZone** class.

CurrentTimeZone

ToString

[C#] public static TimeZone CurrentTimeZone {get;}

[C++] public: __property static TimeZone* get_CurrentTimeZone();

[VB] Public Shared ReadOnly Property CurrentTimeZone As TimeZone

[JScript] public static function get CurrentTimeZone() : TimeZone;

Description

Gets the time zone of the current computer system.

DaylightName

ToString

[C#] public abstract string DaylightName {get;}

[C++] public: __property virtual String* get_DaylightName() = 0;

[VB] MustOverride Public ReadOnly Property DaylightName As String

[JScript] public abstract function get DaylightName() : String;

Description

Gets the daylight saving time zone name.

If daylight saving time is not used in the time zone, an empty string ("") is returned.

StandardName

ToString

[C#] public abstract string StandardName {get;}

1 [C++] public: __property virtual String* get_StandardName() = 0;

2 [VB] MustOverride Public ReadOnly Property StandardName As String

3 [JScript] public abstract function get StandardName() : String;

4
5 *Description*

6 Gets the standard time zone name.

7 GetDaylightChanges

8
9 [C#] public abstract DaylightTime GetDaylightChanges(int year);

10 [C++] public: virtual DaylightTime* GetDaylightChanges(int year) = 0;

11 [VB] MustOverride Public Function GetDaylightChanges(ByVal year As Integer)

12 As DaylightTime

13 [JScript] public abstract function GetDaylightChanges(year : int) : DaylightTime;

14
15 *Description*

16 Returns the daylight saving time period for a particular year.

17 *Return Value:* A **System.Globalization.DaylightTime** instance containing the
18 start and end date for daylight saving time in *year* .

19 Only one daylight saving time period per year is supported. If daylight
20 saving time is not used in the current time zone, **null** is returned. The year to
21 which the daylight saving time period applies.

22 GetUtcOffset

23
24 [C#] public abstract TimeSpan GetUtcOffset(DateTime time);

25 [C++] public: virtual TimeSpan GetUtcOffset(DateTime time) = 0;

1 [VB] MustOverride Public Function GetUtcOffset(ByVal time As DateTime) As
2 TimeSpan

3 [JScript] public abstract function GetUtcOffset(time : DateTime) : TimeSpan;

5 *Description*

6 Returns the coordinated universal time (UTC) offset for the specified local
7 time.

8 *Return Value:* The UTC offset from *time* , measured in ticks.

9 Coordinated universal time (UTC) was previously known as Greenwich
10 Mean Time (GMT). "Local time" is the date and time on the computer you are
11 using. "Offset" is the difference between local time and UTC. That is: local time =
12 UTC + offset *time* must be in the Gregorian calendar and the time zone
13 represented by this instance. If *time* is in daylight saving time, this method returns
14 the UTC offset to the daylight saving time zone. This method obtains the daylight
15 saving time rule from the system. The local date and time.

16 *IsDaylightSavingTime*

17
18 [C#] public virtual bool IsDaylightSavingTime(DateTime time);

19 [C++] public: virtual bool IsDaylightSavingTime(DateTime time);

20 [VB] Overridable Public Function IsDaylightSavingTime(ByVal time As
21 DateTime) As Boolean

22 [JScript] public function IsDaylightSavingTime(time : DateTime) : Boolean;

23 Returns a value indicating whether a specified date and time is within a daylight
24 saving time period.

1
2 *Description*

3 Returns a value indicating whether the specified date and time is within a
4 daylight saving time period.

5 *Return Value:* **true** if *time* is in a daylight saving time period; **false** otherwise, or if
6 *time* is **null** .

7 The year to which the daylight saving time period applies is derived from
8 *time* . A date and time.

9 **IsDaylightSavingTime**

10
11 [C#] public static bool IsDaylightSavingTime(DateTime time, DaylightTime
12 daylightTimes);

13 [C++] public: static bool IsDaylightSavingTime(DateTime time, DaylightTime*
14 daylightTimes);

15 [VB] Public Shared Function IsDaylightSavingTime(ByVal time As DateTime,
16 ByVal daylightTimes As DaylightTime) As Boolean

17 [JScript] public static function IsDaylightSavingTime(time : DateTime,
18 daylightTimes : DaylightTime) : Boolean;

19
20 *Description*

21 Returns a value indicating whether the specified date and time is within the
22 specified daylight saving time period.

23 *Return Value:* **true** if *time* is in *daylightTimes* ; otherwise, **false** . A date and time.
24 A daylight saving time period.

25 **ToLocalTime**

1
2 [C#] public virtual DateTime ToLocalTime(DateTime time);

3 [C++] public: virtual DateTime ToLocalTime(DateTime time);

4 [VB] Overridable Public Function ToLocalTime(ByVal time As DateTime) As
5 DateTime

6 [JScript] public function ToLocalTime(time : DateTime) : DateTime;

7
8 *Description*

9 Returns the local time that corresponds to a specified coordinated universal
10 time (UTC).

11 *Return Value:* A **System.DateTime** instance whose value is the local time that
12 corresponds to *time* .

13 Coordinated universal time (UTC) was previously known as Greenwich
14 Mean Time (GMT). "Local time" is the date and time on the computer you are
15 using. "Offset" is the difference between local time and GMT. That is: local time =
16 UTC + offset A UTC time.

17 ToUniversalTime

18
19 [C#] public virtual DateTime ToUniversalTime(DateTime time);

20 [C++] public: virtual DateTime ToUniversalTime(DateTime time);

21 [VB] Overridable Public Function ToUniversalTime(ByVal time As DateTime)
22 As DateTime

23 [JScript] public function ToUniversalTime(time : DateTime) : DateTime;

24
25 *Description*

Returns the coordinated universal time (UTC) that corresponds to a specified local time.

Return Value: A **System.DateTime** instance whose value is the UTC time that corresponds to *time* .

Coordinated universal time (UTC) was previously known as Greenwich Mean Time (GMT). "Local time" is the date and time on the computer you are using. "Offset" is the difference between local time and UTC. That is: UTC = local time - offset The local date and time.

Type class (System)

ToUniversalTime

Description

Represents type declarations: class types, interface types, array types, value types, and enumeration types.

Type is the root of all reflection operations and the object that represents a type inside the system.

ToUniversalTime

[C#] public static readonly char Delimiter;

[C++] public: static __wchar_t Delimiter;

[VB] Public Shared ReadOnly Delimiter As Char

[JScript] public static var Delimiter : Char;

Description

Separates names in the namespace of the **System.Type** . This field is read-only.

ToUniversalTime

[C#] public static readonly Type[] EmptyTypes;

[C++] public: static Type* EmptyTypes[];

[VB] Public Shared ReadOnly EmptyTypes() As Type

[JScript] public static var EmptyTypes : Type[];

Description

Represents an empty array of type **System.Type** . This field is read-only.

ToUniversalTime

[C#] public static readonly MemberFilter FilterAttribute;

[C++] public: static MemberFilter* FilterAttribute;

[VB] Public Shared ReadOnly FilterAttribute As MemberFilter

[JScript] public static var FilterAttribute : MemberFilter;

Description

Represents the member filter used on attributes. This field is read-only.

This field holds a reference to the delegate used by the

System.Type.FindMembers(System.Reflection.MemberTypes, System.Reflection.BindingFlags, System.Reflection.MemberFilter, System.Object) method.

The method encapsulated by this delegate takes two parameters: the first is a **System.Reflection.MemberInfo** object and the second is an **Object** . The method

determines whether the **MemberInfo** object matches the criteria specified by the **Object** . The **Object** may be assigned the value of any one of the fields on the classes **System.Reflection.FieldAttributes** , **System.Reflection.MethodAttributes** , or **System.Reflection.MethodImplAttributes** .

ToUniversalTime

[C#] public static readonly MemberFilter FilterName;

[C++] public: static MemberFilter* FilterName;

[VB] Public Shared ReadOnly FilterName As MemberFilter

[JScript] public static var FilterName : MemberFilter;

Description

Represents the case-sensitive member filter used on names. This field is read-only.

This field holds a reference to the delegate used by the **System.Type.FindMembers(System.Reflection.MemberTypes, System.Reflection.BindingFlags, System.Reflection.MemberFilter, System.Object)** method.

The method encapsulated by this delegate takes two parameters: the first is a **System.Reflection.MemberInfo** object and the second is an **Object** . The method determines whether the **MemberInfo** object matches the criteria specified by the **Object** . The **Object** is assigned a string value, which may include a trailing "*" wildcard character. Only wildcard end string matching is supported.

ToUniversalTime

1
2 [C#] public static readonly MemberFilter FilterNameIgnoreCase;

3 [C++] public: static MemberFilter* FilterNameIgnoreCase;

4 [VB] Public Shared ReadOnly FilterNameIgnoreCase As MemberFilter

5 [JScript] public static var FilterNameIgnoreCase : MemberFilter;

6
7 *Description*

8 Represents the case-insensitive member filter used on names. This field is
9 read-only.

10 This field holds a reference to the delegate used by the
11 **System.Type.FindMembers(System.Reflection.MemberTypes, System.Reflection.
12 on.BindingFlags, System.Reflection.MemberFilter, System.Object)** method.

13 The method encapsulated by this delegate takes two parameters: the first is a
14 **System.Reflection.MemberInfo** object and the second is an **Object** . The method
15 determines whether the **MemberInfo** object matches the criteria specified by the
16 **Object** . The **Object** is assigned a string value, which may include a trailing "*"
17 wildcard character. Only wildcard end string matching is supported.

18 ToUniversalTime

19
20 [C#] public static readonly object Missing;

21 [C++] public: static Object* Missing;

22 [VB] Public Shared ReadOnly Missing As Object

23 [JScript] public static var Missing : Object;

24
25 *Description*

Represents a missing value in the **System.Type** information. This field is read-only.

Use the **Missing** field to obtain the default value of a parameter. If the **Missing** field is passed in for a parameter value and there is no default value for that parameter, an **System.ArgumentException** is thrown.

Type

Example Syntax:

ToUniversalTime

[C#] protected Type();

[C++] protected: Type();

[VB] Protected Sub New()

[JScript] protected function Type();

Description

Initializes a new instance of the **System.Type** class.

This constructor is invoked by derived classes during the construction of type objects.

Assembly

ToUniversalTime

[C#] public abstract Assembly Assembly {get;}

[C++] public: __property virtual Assembly* get_Assembly() = 0;

[VB] MustOverride Public ReadOnly Property Assembly As Assembly

[JScript] public abstract function get Assembly() : Assembly;

1
2 *Description*

3 Gets the **System.Reflection.Assembly** associated with a class.

4 AssemblyQualifiedName

5 ToUniversalTime

6
7 [C#] public abstract string AssemblyQualifiedName {get;}

8 [C++] public: __property virtual String* get_AssemblyQualifiedName() = 0;

9 [VB] MustOverride Public ReadOnly Property AssemblyQualifiedName As String

10 [JScript] public abstract function get AssemblyQualifiedName() : String;

11
12 *Description*

13 Gets the fully qualified name of the **System.Type** , including the name of
14 the assembly from which the **System.Type** was loaded.

15 All compilers that support the common language runtime will emit the
16 simple name of a nested class, and reflection constructs a mangled name when
17 queried, in accordance with the following conventions.

18 Attributes

19 ToUniversalTime

20
21 [C#] public TypeAttributes Attributes {get;}

22 [C++] public: __property TypeAttributes get_Attributes();

23 [VB] Public ReadOnly Property Attributes As TypeAttributes

24 [JScript] public function get Attributes() : TypeAttributes;

1
2 *Description*

3 Gets the attributes associated with the **System.Type** .

4 BaseType

5 ToUniversalTime

6
7 [C#] public abstract Type BaseType {get;}

8 [C++] public: __property virtual Type* get_BaseType() = 0;

9 [VB] MustOverride Public ReadOnly Property BaseType As Type

10 [JScript] public abstract function get BaseType() : Type;

11
12 *Description*

13 Gets the type from which the current **System.Type** directly inherits.

14 The base type is the type from which the current type directly inherits.

15 DeclaringType

16 ToUniversalTime

17
18 [C#] public override Type DeclaringType {get;}

19 [C++] public: __property virtual Type* get_DeclaringType();

20 [VB] Overrides Public ReadOnly Property DeclaringType As Type

21 [JScript] public function get DeclaringType() : Type;

22
23 *Description*

24 Gets the class that declares this member.

This property implements the abstract property inherited from
System.Reflection.MemberInfo .

DefaultBinder

ToUniversalTime

[C#] public static Binder DefaultBinder {get;}

[C++] public: __property static Binder* get_DefaultBinder();

[VB] Public Shared ReadOnly Property DefaultBinder As Binder

[JScript] public static function get DefaultBinder() : Binder;

Description

Gets the default binder used by the system.

Reflection models the accessibility rules of the common type system. For example, if the caller is in the same assembly, the caller does not need special permissions for internal members. Otherwise, the caller needs

System.Security.Permissions.ReflectionPermission . This is consistent with lookup of members that are protected, private, and so on.

FullName

ToUniversalTime

[C#] public abstract string FullName {get;}

[C++] public: __property virtual String* get_FullName() = 0;

[VB] MustOverride Public ReadOnly Property FullName As String

[JScript] public abstract function get FullName() : String;

1
2 *Description*

3 Gets the fully qualified name of the **System.Type** , including the
4 namespace of the **System.Type** .

5 All compilers that support the common language runtime will emit the
6 simple name of a nested class, and reflection constructs a mangled name when
7 queried, in accordance with the following conventions.

8 GUID

9 ToUniversalTime

10
11 [C#] public abstract Guid GUID {get;}

12 [C++] public: __property virtual Guid get_GUID() = 0;

13 [VB] MustOverride Public ReadOnly Property GUID As Guid

14 [JScript] public abstract function get GUID() : Guid;

15
16 *Description*

17 Gets the GUID associated with the **System.Type** .

18 HasElementType

19 ToUniversalTime

20
21 [C#] public bool HasElementType {get;}

22 [C++] public: __property bool get_HasElementType();

23 [VB] Public ReadOnly Property HasElementType As Boolean

24 [JScript] public function get HasElementType() : Boolean;

Description

Gets a value indicating whether the current **System.Type** encompasses or refers to another type; that is, whether the current **System.Type** is an array, a pointer, or is passed by reference.

For example, `Type.GetType("Int32[]").HasElementType` returns **true** , but `Type.GetType("Int32").HasElementType` returns **false** . `HasElementType` also returns **true** for `"Int32*"` and `"Int32&"`.

IsAbstract

ToUniversalTime

[C#] public bool IsAbstract {get;}

[C++] public: __property bool get_IsAbstract();

[VB] Public ReadOnly Property IsAbstract As Boolean

[JScript] public function get IsAbstract() : Boolean;

Description

Gets a value indicating whether the **System.Type** is abstract and must be overridden.

IsAnsiClass

ToUniversalTime

[C#] public bool IsAnsiClass {get;}

[C++] public: __property bool get_IsAnsiClass();

[VB] Public ReadOnly Property IsAnsiClass As Boolean

1 [JScript] public function get IsAnsiClass() : Boolean;

3 *Description*

4 Gets a value indicating whether the string format attribute **AnsiClass** is
5 selected for the **System.Type** .

6 The **System.Reflection.TypeAttributes.StringFormatMask** selects the
7 string format attributes. The string format attributes enhance interoperability by
8 defining how strings should be interpreted.

9 IsArray

10 ToUniversalTime

12 [C#] public bool IsArray {get;}

13 [C++] public: __property bool get_IsArray();

14 [VB] Public ReadOnly Property IsArray As Boolean

15 [JScript] public function get IsArray() : Boolean;

17 *Description*

18 Gets a value indicating whether the **System.Type** is an array.

19 An instance of the **System.Array** class will return **false** because it is an
20 object, not an array.

21 IsAutoClass

22 ToUniversalTime

24 [C#] public bool IsAutoClass {get;}

25 [C++] public: __property bool get_IsAutoClass();

1 [VB] Public ReadOnly Property IsAutoClass As Boolean

2 [JScript] public function get IsAutoClass() : Boolean;

3
4 *Description*

5 Gets a value indicating whether the string format attribute **AutoClass** is
6 selected for the **System.Type** .

7 The **System.Reflection.TypeAttributes.StringFormatMask** selects the
8 string format attributes. The string format attributes enhance interoperability by
9 defining how strings should be interpreted.

10 IsAutoLayout

11 ToUniversalTime

12
13 [C#] public bool IsAutoLayout {get;}

14 [C++] public: __property bool get_IsAutoLayout();

15 [VB] Public ReadOnly Property IsAutoLayout As Boolean

16 [JScript] public function get IsAutoLayout() : Boolean;

17
18 *Description*

19 Gets a value indicating whether the class layout attribute **AutoLayout** is
20 selected for the **System.Type** .

21 The **System.Reflection.TypeAttributes.LayoutMask** is used to select the
22 class layout attributes. The class layout attributes (
23 **AutoLayout**, **SequentialLayout** and **ExplicitLayout**) define how the fields of the
24 class instance are laid out in memory.

25 IsByRef

ToUniversalTime

[C#] public bool IsByRef {get;}

[C++] public: __property bool get_IsByRef();

[VB] Public ReadOnly Property IsByRef As Boolean

[JScript] public function get IsByRef() : Boolean;

Description

Gets a value indicating whether the **System.Type** is passed by reference.

IsClass

ToUniversalTime

[C#] public bool IsClass {get;}

[C++] public: __property bool get_IsClass();

[VB] Public ReadOnly Property IsClass As Boolean

[JScript] public function get IsClass() : Boolean;

Description

Gets a value indicating whether the **System.Type** is a class; that is, not a value type or interface.

The **System.Reflection.TypeAttributes.ClassSemanticsMask** distinguishes a type declaration as class, interface, or value type.

IsCOMObject

ToUniversalTime

1
2 [C#] public bool IsCOMObject {get;}

3 [C++] public: __property bool get_IsCOMObject();

4 [VB] Public ReadOnly Property IsCOMObject As Boolean

5 [JScript] public function get IsCOMObject() : Boolean;

6
7 *Description*

8 Gets a value indicating whether the **System.Type** is a COM object.

9 This method returns **false** for COM interfaces because they are not objects.

10 COM interfaces can be implemented by Microsoft .NET Framework objects.

11 IsContextful

12 ToUniversalTime

13
14 [C#] public bool IsContextful {get;}

15 [C++] public: __property bool get_IsContextful();

16 [VB] Public ReadOnly Property IsContextful As Boolean

17 [JScript] public function get IsContextful() : Boolean;

18
19 *Description*

20 Gets a value indicating whether the **System.Type** can be hosted in a
21 context.

22 A context intercepts calls to the class members and enforces policies that
23 are applied to the class, such as synchronization. For more detailed information on
24 remoting contexts, see **System.Runtime.Remoting.Contexts.Context** .

25 IsEnum

ToUniversalTime

[C#] public bool IsEnum {get;}

[C++] public: __property bool get_IsEnum();

[VB] Public ReadOnly Property IsEnum As Boolean

[JScript] public function get IsEnum() : Boolean;

Description

Gets a value indicating whether the **System.Type** is an enumeration.

For example, GetType(Enum).IsEnum() returns **false** because

System.Enum is an object, not an enumeration.

IsExplicitLayout

ToUniversalTime

[C#] public bool IsExplicitLayout {get;}

[C++] public: __property bool get_IsExplicitLayout();

[VB] Public ReadOnly Property IsExplicitLayout As Boolean

[JScript] public function get IsExplicitLayout() : Boolean;

Description

Gets a value indicating whether the class layout attribute **ExplicitLayout** is selected for the **System.Type** .

The **System.Reflection.TypeAttributes.LayoutMask** is used to select the class layout attributes. The class layout attributes (

AutoLayout, SequentialLayout and ExplicitLayout) define how the fields of the class instance are laid out in memory.

IsImport

ToUniversalTime

[C#] public bool IsImport {get;}

[C++] public: __property bool get_IsImport();

[VB] Public ReadOnly Property IsImport As Boolean

[JScript] public function get IsImport() : Boolean;

Description

Gets a value indicating whether the **System.Type** was imported from another class.

IsInterface

ToUniversalTime

[C#] public bool IsInterface {get;}

[C++] public: __property bool get_IsInterface();

[VB] Public ReadOnly Property IsInterface As Boolean

[JScript] public function get IsInterface() : Boolean;

Description

Gets a value indicating whether the **System.Type** is an interface; that is, not a class or a value type.

The **System.Reflection.TypeAttributes.ClassSemanticsMask**

distinguishes a type declaration as class, interface or value type.

IsLayoutSequential

ToUniversalTime

[C#] public bool IsLayoutSequential {get;}

[C++] public: __property bool get_IsLayoutSequential();

[VB] Public ReadOnly Property IsLayoutSequential As Boolean

[JScript] public function get IsLayoutSequential() : Boolean;

Description

Gets a value indicating whether the class layout attribute

SequentialLayout is selected for the **System.Type** .

The **System.Reflection.TypeAttributes.LayoutMask** is used to select the class layout attributes. The class layout attributes (**AutoLayout**, **SequentialLayout** and **ExplicitLayout**) define how the fields of the class instance are laid out in memory.

IsMarshalByRef

ToUniversalTime

[C#] public bool IsMarshalByRef {get;}

[C++] public: __property bool get_IsMarshalByRef();

[VB] Public ReadOnly Property IsMarshalByRef As Boolean

[JScript] public function get IsMarshalByRef() : Boolean;

1
2 *Description*

3 Gets a value indicating whether the Type is marshaled by reference.

4 IsNestedAssembly

5 ToUniversalTime

6
7 [C#] public bool IsNestedAssembly {get;}

8 [C++] public: __property bool get_IsNestedAssembly();

9 [VB] Public ReadOnly Property IsNestedAssembly As Boolean

10 [JScript] public function get IsNestedAssembly() : Boolean;

11
12 *Description*

13 Gets a value indicating whether the **System.Type** is nested and visible only
14 within its own assembly.

15 The **System.Reflection.TypeAttributes.VisibilityMask** selects the
16 visibility attributes.

17 IsNestedFamANDAssem

18 ToUniversalTime

19
20 [C#] public bool IsNestedFamANDAssem {get;}

21 [C++] public: __property bool get_IsNestedFamANDAssem();

22 [VB] Public ReadOnly Property IsNestedFamANDAssem As Boolean

23 [JScript] public function get IsNestedFamANDAssem() : Boolean;

24
25 *Description*

Gets a value indicating whether the **System.Type** is nested and visible only to classes that belong to both its own family and its own assembly.

The **System.Reflection.TypeAttributes.VisibilityMask** selects the visibility attributes.

IsNestedFamily

ToUniversalTime

[C#] public bool IsNestedFamily {get;}

[C++] public: __property bool get_IsNestedFamily();

[VB] Public ReadOnly Property IsNestedFamily As Boolean

[JScript] public function get IsNestedFamily() : Boolean;

Description

Gets a value indicating whether the **System.Type** is nested and visible only within its own family.

The **System.Reflection.TypeAttributes.VisibilityMask** selects the visibility attributes.

IsNestedFamORAssem

ToUniversalTime

[C#] public bool IsNestedFamORAssem {get;}

[C++] public: __property bool get_IsNestedFamORAssem();

[VB] Public ReadOnly Property IsNestedFamORAssem As Boolean

[JScript] public function get IsNestedFamORAssem() : Boolean;

1
2 *Description*

3 Gets a value indicating whether the **System.Type** is nested and visible only
4 to classes that belong to either its own family or to its own assembly.

5 The **System.Reflection.TypeAttributes.VisibilityMask** selects the
6 visibility attributes.

7 IsNestedPrivate

8 ToUniversalTime

9
10 [C#] public bool IsNestedPrivate {get;}

11 [C++] public: __property bool get_IsNestedPrivate();

12 [VB] Public ReadOnly Property IsNestedPrivate As Boolean

13 [JScript] public function get IsNestedPrivate() : Boolean;

14
15 *Description*

16 Gets a value indicating whether the **System.Type** is nested and declared
17 private.

18 The **System.Reflection.TypeAttributes.VisibilityMask** selects the
19 visibility attributes.

20 IsNestedPublic

21 ToUniversalTime

22
23 [C#] public bool IsNestedPublic {get;}

24 [C++] public: __property bool get_IsNestedPublic();

25 [VB] Public ReadOnly Property IsNestedPublic As Boolean

1 [JScript] public function get IsNestedPublic() : Boolean;

3 *Description*

4 Gets a value indicating whether the **System.Type** is nested and declared
5 public.

6 The **System.Reflection.TypeAttributes.VisibilityMask** selects the
7 visibility attributes.

8 IsNotPublic

9 ToUniversalTime

11 [C#] public bool IsNotPublic {get;}

12 [C++] public: __property bool get_IsNotPublic();

13 [VB] Public ReadOnly Property IsNotPublic As Boolean

14 [JScript] public function get IsNotPublic() : Boolean;

16 *Description*

17 Gets a value indicating whether the top-level **System.Type** is not declared
18 public.

19 **IsPublic** and **IsNotPublic** get the visibility of the top-level type only.

20 IsPointer

21 ToUniversalTime

23 [C#] public bool IsPointer {get;}

24 [C++] public: __property bool get_IsPointer();

25 [VB] Public ReadOnly Property IsPointer As Boolean

1 [JScript] public function get IsPointer() : Boolean;

3 *Description*

4 Gets a value indicating whether the **System.Type** is a pointer.

5 IsPrimitive

6 ToUniversalTime

8 [C#] public bool IsPrimitive {get;}

9 [C++] public: __property bool get_IsPrimitive();

10 [VB] Public ReadOnly Property IsPrimitive As Boolean

11 [JScript] public function get IsPrimitive() : Boolean;

13 *Description*

14 Gets a value indicating whether the **System.Type** is one of the primitive
15 types.

16 The primitive types are **System.Boolean** , **System.Byte** , **System.SByte** ,
17 **System.Int16** , **System.UInt16** , **System.Int32** , **System.UInt32** , **System.Int64** ,
18 **System.UInt64** , **System.Char** , **System.Double** , and **System.Single** .

19 IsPublic

20 ToUniversalTime

22 [C#] public bool IsPublic {get;}

23 [C++] public: __property bool get_IsPublic();

24 [VB] Public ReadOnly Property IsPublic As Boolean

25 [JScript] public function get IsPublic() : Boolean;

1
2 *Description*

3 Gets a value indicating whether the top-level **System.Type** is declared
4 public.

5 **IsPublic** and **IsNotPublic** get the visibility of the top-level type only.

6 **IsSealed**

7 **ToUniversalTime**

8
9 [C#] public bool IsSealed {get;}

10 [C++] public: __property bool get_IsSealed();

11 [VB] Public ReadOnly Property IsSealed As Boolean

12 [JScript] public function get IsSealed() : Boolean;

13
14 *Description*

15 Gets a value indicating whether the **System.Type** is declared sealed.

16 **IsSerializable**

17 **ToUniversalTime**

18
19 [C#] public bool IsSerializable {get;}

20 [C++] public: __property bool get_IsSerializable();

21 [VB] Public ReadOnly Property IsSerializable As Boolean

22 [JScript] public function get IsSerializable() : Boolean;

23
24 *Description*

25 Gets a value indicating whether the **System.Type** is serializable.

IsSpecialName

ToUniversalTime

[C#] public bool IsSpecialName {get;}

[C++] public: __property bool get_IsSpecialName();

[VB] Public ReadOnly Property IsSpecialName As Boolean

[JScript] public function get IsSpecialName() : Boolean;

Description

Gets a value indicating whether the **System.Type** has a name that requires special handling.

Names that begin with or contain an underscore character (`_`), property accessors, and operator overloading methods are examples of types that might require special treatment by some compilers.

IsUnicodeClass

ToUniversalTime

[C#] public bool IsUnicodeClass {get;}

[C++] public: __property bool get_IsUnicodeClass();

[VB] Public ReadOnly Property IsUnicodeClass As Boolean

[JScript] public function get IsUnicodeClass() : Boolean;

Description

Gets a value indicating whether the string format attribute **UnicodeClass** is selected for the **System.Type**.

The **System.Reflection.TypeAttributes.StringFormatMask** is used to select the string format attributes. The string format attributes enhance interoperability by defining how strings should be interpreted.

IsValueType

ToUniversalTime

[C#] public bool IsValueType {get;}

[C++] public: __property bool get_IsValueType();

[VB] Public ReadOnly Property IsValueType As Boolean

[JScript] public function get IsValueType() : Boolean;

Description

Gets a value indicating whether the **System.Type** is a value type.

Value types are those that are represented as sequences of bits; value types are not classes or interfaces. These are referred to as "structs" in some programming languages. Enums are a special case of value types.

MemberType

ToUniversalTime

[C#] public override MemberTypes MemberType {get;}

[C++] public: __property virtual MemberTypes get_MemberType();

[VB] Overrides Public ReadOnly Property MemberType As MemberTypes

[JScript] public function get MemberType() : MemberTypes;

Description

Gets a bitmask indicating the member type.

Module

ToUniversalTime

[C#] public abstract Module Module {get;}

[C++] public: __property virtual Module* get_Module() = 0;

[VB] MustOverride Public ReadOnly Property Module As Module

[JScript] public abstract function get Module() : Module;

Description

Gets the module (the DLL) in which the current **System.Type** is defined.

Name

Namespace

ToUniversalTime

Description

Gets the namespace of the **System.Type** .

ReflectedType

ToUniversalTime

[C#] public override Type ReflectedType {get;}

[C++] public: __property virtual Type* get_ReflectedType();

[VB] Overrides Public ReadOnly Property ReflectedType As Type

[JScript] public function get ReflectedType() : Type;

1
2 *Description*

3 Gets the class object that was used to obtain this member.

4 This property implements the abstract property inherited from

5 **System.Reflection.MemberInfo** .

6 TypeHandle

7 ToUniversalTime

8
9 [C#] public abstract RuntimeTypeHandle TypeHandle {get;}

10 [C++] public: __property virtual RuntimeTypeHandle get_TypeHandle() = 0;

11 [VB] MustOverride Public ReadOnly Property TypeHandle As

12 RuntimeTypeHandle

13 [JScript] public abstract function get TypeHandle() : RuntimeTypeHandle;

14
15 *Description*

16 Gets the handle for the current **System.Type** .

17 **TypeHandle** encapsulates a pointer to an internal data structure that
18 represents the type. This handle is unique during the process lifetime. The handle
19 is valid only in the application domain in which it was obtained.

20 TypeInitializer

21 ToUniversalTime

22
23 [C#] public ConstructorInfo TypeInitializer {get;}

24 [C++] public: __property ConstructorInfo* get_TypeInitializer();

25 [VB] Public ReadOnly Property TypeInitializer As ConstructorInfo

1 [JScript] public function get TypeInitializer() : ConstructorInfo;

3 *Description*

4 Gets the name of the class constructor for the **System.Type** .

5 Class initializers are available through

6 **System.Type.GetMember(System.String)** , **System.Type.GetMembers** ,
7 **System.Type.FindMembers(System.Reflection.MemberTypes, System.Reflection**
8 **on.BindingFlags, System.Reflection.MemberFilter, System.Object)** , and
9 **System.Type.GetConstructors** .

10 UnderlyingSystemType

11 ToUniversalTime

13 [C#] public abstract Type UnderlyingSystemType {get;}

14 [C++] public: __property virtual Type* get_UnderlyingSystemType() = 0;

15 [VB] MustOverride Public ReadOnly Property UnderlyingSystemType As Type

16 [JScript] public abstract function get UnderlyingSystemType() : Type;

18 *Description*

19 Indicates the type provided by the common language runtime that
20 represents this type.

21 Equals

23 [C#] public override bool Equals(object o);

24 [C++] public: bool Equals(Object* o);

25 [VB] Overrides Public Function Equals(ByVal o As Object) As Boolean

1 [JScript] public override function Equals(o : Object) : Boolean; Determines if the
2 underlying system type of the current **System.Type** is the same as the underlying
3 system type of the specified **System.Object** or **System.Type** .

4 *Description*

6 Determines if the underlying system type of the current **System.Type** is the
7 same as the underlying system type of the specified **System.Object** .

8 *Return Value:* **true** if the underlying system type of *o* is the same as the
9 underlying system type of the current **System.Type** ; otherwise, **false** . The
10 **System.Object** whose underlying system type is to be compared with the
11 underlying system type of the current **System.Type**.

12 *Equals*

13
14 [C#] public new bool Equals(Type o);

15 [C++] public: bool Equals(Type* o);

16 [VB] Shadows Public Function Equals(ByVal o As Type) As Boolean

17 [JScript] public hide function Equals(o : Type) : Boolean;

18 19 *Description*

20 Determines if the underlying system type of the current **System.Type** is the
21 same as the underlying system type of the specified **System.Type** .

22 *Return Value:* **true** if the underlying system type of *o* is the same as the
23 underlying system type of the current **System.Type** ; otherwise, **false** . The
24 **System.Type** whose underlying system type is to be compared with the
25 underlying system type of the current **System.Type**.

FindInterfaces

```
[C#] public virtual Type[] FindInterfaces(TypeFilter filter, object filterCriteria);  
[C++] public: virtual Type* FindInterfaces(TypeFilter* filter, Object*  
filterCriteria) [];  
[VB] Overridable Public Function FindInterfaces(ByVal filter As TypeFilter,  
ByVal filterCriteria As Object) As Type()  
[JScript] public function FindInterfaces(filter : TypeFilter, filterCriteria : Object) :  
Type[];
```

Description

Returns an array of **System.Type** objects representing a filtered list of interfaces implemented or inherited by the current **System.Type**.

Return Value: An array of **System.Type** objects representing a filtered list of the interfaces implemented or inherited by the current **System.Type**.

This method can be overridden by a derived class. The **System.Reflection.TypeFilter** delegate that compares the interfaces against *filterCriteria*. The search criteria that determines whether an interface should be included in the returned array.

FindMembers

```
[C#] public virtual MemberInfo[] FindMembers(MemberTypes memberType,  
BindingFlags bindingAttr, MemberFilter filter, object filterCriteria);  
[C++] public: virtual MemberInfo* FindMembers(MemberTypes memberType,  
BindingFlags bindingAttr, MemberFilter* filter, Object* filterCriteria) [];
```

```

1 [VB] Overridable Public Function FindMembers(ByVal memberType As
2 MemberTypes, ByVal bindingAttr As BindingFlags, ByVal filter As
3 MemberFilter, ByVal filterCriteria As Object) As MemberInfo()
4 [JScript] public function FindMembers(memberType : MemberTypes, bindingAttr
5 : BindingFlags, filter : MemberFilter, filterCriteria : Object) : MemberInfo[];
6

```

Description

Returns a filtered array of **System.Reflection.MemberInfo** objects of the specified member type.

Return Value: A filtered array of **System.Reflection.MemberInfo** objects of the specified member type.

This method can be overridden by a derived class. A **MemberTypes** object indicating the type of member to search for. A bitmask comprised of one or more **System.Reflection.BindingFlags** that specify how the search is conducted. The delegate that does the comparisons, returning **true** if the member currently being inspected matches the *filterCriteria* and **false** otherwise. You can use the **FilterAttribute**, **FilterName**, and **FilterNameIgnoreCase** delegates supplied by this class. The first uses the fields of **FieldAttributes**, **MethodAttributes**, and **MethodImplAttributes** as search criteria, and the other two delegates use **String** objects as the search criteria. The search criteria that determines whether a member is returned in the array of **MemberInfo** objects.

GetArrayRank

```

24 [C#] public virtual int GetArrayRank();
25 [C++] public: virtual int GetArrayRank();

```

1 [VB] Overridable Public Function GetArrayRank() As Integer

2 [JScript] public function GetArrayRank() : int;

3
4 *Description*

5 Gets the number of dimensions in an **System.Array** .

6 *Return Value:* The number of dimensions in an **System.Array** .

7 **GetAttributeFlagsImpl**

8
9 [C#] protected abstract TypeAttributes GetAttributeFlagsImpl();

10 [C++] protected: virtual TypeAttributes GetAttributeFlagsImpl() = 0;

11 [VB] MustOverride Protected Function GetAttributeFlagsImpl() As

12 TypeAttributes

13 [JScript] protected abstract function GetAttributeFlagsImpl() : TypeAttributes;

14
15 *Description*

16 When overridden in a derived class, implements the
17 **System.Type.Attributes** property and gets a bitmask indicating the attributes
18 associated with the **System.Type** .

19 *Return Value:* A **System.Reflection.TypeAttributes** object representing the
20 attribute set of the **System.Type** .

21 **GetConstructor**

22
23 [C#] public ConstructorInfo GetConstructor(Type[] types);

24 [C++] public: ConstructorInfo* GetConstructor(Type* types[]);

25 [VB] Public Function GetConstructor(ByVal types() As Type) As ConstructorInfo

1 [JScript] public function GetConstructor(types : Type[]) : ConstructorInfo;

3 *Description*

4 Searches for a public instance constructor whose parameters match the
5 types in the specified array.

6 *Return Value:* A **System.Reflection.ConstructorInfo** object representing the
7 public instance constructor whose parameters match the types in the parameter
8 type array, if found; otherwise, **null** .

9 **System.Type.GetConstructor(System.Reflection.BindingFlags, System.
10 Reflection.Binder, System.Reflection.CallingConventions, System.Type[], System.
11 m.Reflection.ParameterModifier[])** looks for public instance constructors and
12 cannot be used to obtain a class initializer. Class initializers are available through
13 **System.Type.GetMember(System.String)** , **System.Type.GetMembers** ,
14 **System.Type.FindMembers(System.Reflection.MemberTypes, System.Reflection.
15 on.BindingFlags, System.Reflection.MemberFilter, System.Object)** ,
16 **System.Type.GetConstructors** , and **System.Type.TypeInitializer** . An array of
17 **System.Type** objects representing the number, order, and type of the parameters
18 for the constructor to get.

19 **GetConstructor**

20
21 [C#] public ConstructorInfo GetConstructor(BindingFlags bindingAttr, Binder
22 binder, Type[] types, ParameterModifier[] modifiers);

23 [C++] public: ConstructorInfo* GetConstructor(BindingFlags bindingAttr,
24 Binder* binder, Type* types[], ParameterModifier modifiers[]);

25 [VB] Public Function GetConstructor(ByVal bindingAttr As BindingFlags, ByVal

```

1 binder As Binder, ByVal types() As Type, ByVal modifiers() As
2 ParameterModifier) As ConstructorInfo
3 [JScript] public function GetConstructor(bindingAttr : BindingFlags, binder :
4 Binder, types : Type[], modifiers : ParameterModifier[]) : ConstructorInfo;
5

```

Description

Searches for a constructor whose parameters match the specified argument types and modifiers, using the specified binding constraints.

Return Value: A **System.Reflection.ConstructorInfo** object representing the constructor that matches the specified requirements, if found; otherwise, **null**.

The *types* array and the *modifiers* array have the same length. A parameter specified in the *types* array can have the following attributes, which are specified in the *modifiers* array: *pdIn*, *pdOut*, *pdLcid*, *pdRetVal*, *pdOptional*, and *pdHasDefault*, which represent [In], [Out], [lcid], [retval], [optional], and a value specifying whether the parameter has a default value. A parameter's associated attributes are stored in the metadata and enhance interoperability. A bitmask comprised of one or more **System.Reflection.BindingFlags** that specify how the search is conducted. A **System.Reflection.Binder** object that defines a set of properties and enables binding, which can involve selection of an overloaded method, coercion of argument types, and invocation of a member through reflection. An array of **System.Type** objects representing the number, order, and type of the parameters for the constructor to get. An array of **System.Reflection.ParameterModifier** objects representing the attributes associated with the corresponding element in the parameter type array.

GetConstructor


```

1
2 [C#] public ConstructorInfo GetConstructor(BindingFlags bindingAttr, Binder
3 binder, CallingConventions callConvention, Type[] types, ParameterModifier[]
4 modifiers);
5 [C++] public: ConstructorInfo* GetConstructor(BindingFlags bindingAttr,
6 Binder* binder, CallingConventions callConvention, Type* types[],
7 ParameterModifier modifiers[]);
8 [VB] Public Function GetConstructor(ByVal bindingAttr As BindingFlags, ByVal
9 binder As Binder, ByVal callConvention As CallingConventions, ByVal types()
10 As Type, ByVal modifiers() As ParameterModifier) As ConstructorInfo
11 [JScript] public function GetConstructor(bindingAttr : BindingFlags, binder :
12 Binder, callConvention : CallingConventions, types : Type[], modifiers :
13 ParameterModifier[]) : ConstructorInfo; Gets a specific constructor of the current
14 System.Type .
15

```

Description

Searches for a constructor whose parameters match the specified argument types and modifiers, using the specified binding constraints and the specified calling convention.

Return Value: A **System.Reflection.ConstructorInfo** object representing the constructor that matches the specified requirements, if found; otherwise, **null** .

The *types* array and the *modifiers* array have the same length. A parameter specified in the *types* array can have the following attributes, which are specified in the *modifiers* array: *pdIn*, *pdOut*, *pdLcid*, *pdRetVal*, *pdOptional*, and *pdHasDefault*, which represent [In], [Out], [lcid], [retval], [optional], and a value

specifying whether the parameter has a default value. A parameter's associated attributes are stored in the metadata and enhance interoperability. A bitmask comprised of one or more **System.Reflection.BindingFlags** that specify how the search is conducted. A **System.Reflection.Binder** object that defines a set of properties and enables binding, which can involve selection of an overloaded method, coercion of argument types, and invocation of a member through reflection. The **System.Reflection.CallingConventions** object that specifies the set of rules to use regarding the order and layout of arguments, how the return value is passed, what registers are used for arguments, and the stack is cleaned up. An array of **System.Type** objects representing the number, order, and type of the parameters for the constructor to get. An array of **System.Reflection.ParameterModifier** objects representing the attributes associated with the corresponding element in the *types* array.

GetConstructorImpl

[C#] protected abstract ConstructorInfo GetConstructorImpl(BindingFlags bindingAttr, Binder binder, CallingConventions callConvention, Type[] types, ParameterModifier[] modifiers);
 [C++] protected: virtual ConstructorInfo* GetConstructorImpl(BindingFlags bindingAttr, Binder* binder, CallingConventions callConvention, Type* types[], ParameterModifier modifiers[]) = 0;
 [VB] MustOverride Protected Function GetConstructorImpl(ByVal bindingAttr As BindingFlags, ByVal binder As Binder, ByVal callConvention As CallingConventions, ByVal types() As Type, ByVal modifiers() As ParameterModifier) As ConstructorInfo

```

1 [JScript] protected abstract function GetConstructorImpl(bindingAttr :
2 BindingFlags, binder : Binder, callConvention : CallingConventions, types :
3 Type[], modifiers : ParameterModifier[]) : ConstructorInfo;
4

```

Description

When overridden in a derived class, searches for a constructor whose parameters match the specified argument types and modifiers, using the specified binding constraints and the specified calling convention.

Return Value: A **System.Reflection.ConstructorInfo** object representing the constructor that matches the specified requirements, if found; otherwise, **null**.

The *types* array and the *modifiers* array have the same length. A parameter specified in the *types* array can have the following attributes, which are specified in the *modifiers* array: *pdIn*, *pdOut*, *pdLcid*, *pdRetval*, *pdOptional*, and *pdHasDefault*, which represent [In], [Out], [lcid], [retval], [optional], and a value specifying whether the parameter has a default value. A parameter's associated attributes are stored in the metadata and are used for interoperability. A bitmask comprised of one or more **System.Reflection.BindingFlags** that specify how the search is conducted. A **System.Reflection.Binder** object that defines a set of properties and enables binding, which can involve selection of an overloaded method, coercion of argument types, and invocation of a member through reflection. The **System.Reflection.CallingConventions** object that specifies the set of rules to use regarding the order and layout of arguments, how the return value is passed, what registers are used for arguments, and the stack is cleaned up. An array of **System.Type** objects representing the number, order, and type of the parameters for the constructor to get. An array of

System.Reflection.ParameterModifier objects representing the attributes associated with the corresponding element in the *types* array.

GetConstructors

[C#] public ConstructorInfo[] GetConstructors();

[C++] public: ConstructorInfo* GetConstructors() [];

[VB] Public Function GetConstructors() As ConstructorInfo()

[JScript] public function GetConstructors() : ConstructorInfo[]; Gets the constructors of the current **System.Type** .

Description

Returns all the public constructors defined for the current **System.Type** .

Return Value: An array of **System.Reflection.ConstructorInfo** objects representing all the public constructors defined for the current **System.Type** , including the type initializer if it is defined.

The following table shows what members of a base class are returned by the **Get** methods when reflecting on a type.

GetConstructors

[C#] public abstract ConstructorInfo[] GetConstructors(BindingFlags bindingAttr);

[C++] public: virtual ConstructorInfo* GetConstructors(BindingFlags bindingAttr) [] = 0;

[VB] MustOverride Public Function GetConstructors(ByVal bindingAttr As BindingFlags) As ConstructorInfo()

1 [JScript] public abstract function GetConstructors(bindingAttr : BindingFlags) :
2 ConstructorInfo[];

3
4 *Description*

5 When overridden in a derived class, searches for the constructors defined
6 for the current **System.Type** , using the specified **BindingFlags** .

7 *Return Value:* An array of **System.Reflection.ConstructorInfo** objects
8 representing all constructors defined for the current **System.Type** that match the
9 specified binding constraints, including the type initializer if it is defined.

10 *bindingAttr* can be used to specify whether to return only public
11 constructors or both public and non-public constructors. A bitmask comprised of
12 one or more **System.Reflection.BindingFlags** that specify how the search is
13 conducted.

14 **GetDefaultMembers**

15
16 [C#] public virtual MemberInfo[] GetDefaultMembers();

17 [C++] public: virtual MemberInfo* GetDefaultMembers() [];

18 [VB] Overridable Public Function GetDefaultMembers() As MemberInfo()

19 [JScript] public function GetDefaultMembers() : MemberInfo[];

20
21 *Description*

22 Searches for the members defined for the current **System.Type** whose
23 **System.Reflection.DefaultMemberAttribute** is set.

24 *Return Value:* An array of **System.Reflection.MemberInfo** objects representing
25 all default members of the current **System.Type** .

This method can be overridden by a derived class.

GetElementType

[C#] public abstract Type GetElementType();

[C++] public: virtual Type* GetElementType() = 0;

[VB] MustOverride Public Function GetElementType() As Type

[JScript] public abstract function GetElementType() : Type;

Description

When overridden in a derived class, returns the **System.Type** of the object encompassed or referred to by the current array, pointer or reference type.

Return Value: The **System.Type** of the object encompassed or referred to by the current array, pointer or reference type.

For example, `Type.GetType("Int32[]").GetElementType` returns `Int32`.

GetEvent

[C#] public EventInfo GetEvent(string name);

[C++] public: EventInfo* GetEvent(String* name);

[VB] Public Function GetEvent(ByVal name As String) As EventInfo

[JScript] public function GetEvent(name : String) : EventInfo; Gets a specific event declared or inherited by the current **System.Type**.

Description

Returns the **System.Reflection.EventInfo** object representing the specified event.

Return Value: The **System.Reflection.EventInfo** object representing the specified event which is declared or inherited by the current **System.Type** , if found; otherwise, **null** .

The search for *name* is case-sensitive. The **System.String** containing the name of an event which is declared or inherited by the current **System.Type**.

GetEvent

[C#] public abstract EventInfo GetEvent(string name, BindingFlags bindingAttr);

[C++] public: virtual EventInfo* GetEvent(String* name, BindingFlags bindingAttr) = 0;

[VB] MustOverride Public Function GetEvent(ByVal name As String, ByVal bindingAttr As BindingFlags) As EventInfo

[JScript] public abstract function GetEvent(name : String, bindingAttr : BindingFlags) : EventInfo;

Description

When overridden in a derived class, returns the **System.Reflection.EventInfo** object representing the specified event, using the specified binding constraints.

Return Value: The **System.Reflection.EventInfo** object representing the specified event which is declared or inherited by the current **System.Type** , if found; otherwise, **null** .

The following **System.Reflection.BindingFlags** filter flags can be used to define which events to include in the search: *Public* to include public events in the search. The **System.String** containing the name of an event which is declared or

1 inherited by the current **System.Type**. A bitmask comprised of one or more
2 **System.Reflection.BindingFlags** that specify how the search is conducted.

3 GetEvents

4
5 [C#] public virtual EventInfo[] GetEvents();

6 [C++] public: virtual EventInfo* GetEvents() [];

7 [VB] Overridable Public Function GetEvents() As EventInfo()

8 [JScript] public function GetEvents() : EventInfo[]; Gets the events that are
9 declared or inherited by the current **System.Type** .

10
11 *Description*

12 Returns all the public events that are declared or inherited by the current
13 **System.Type** .

14 *Return Value:* An array of **System.Reflection.EventInfo** objects representing all
15 the public events which are declared or inherited by the current **System.Type** .

16 This method can be overridden by a derived class.

17 GetEvents

18
19 [C#] public abstract EventInfo[] GetEvents(BindingFlags bindingAttr);

20 [C++] public: virtual EventInfo* GetEvents(BindingFlags bindingAttr) [] = 0;

21 [VB] MustOverride Public Function GetEvents(ByVal bindingAttr As
22 BindingFlags) As EventInfo()

23 [JScript] public abstract function GetEvents(bindingAttr : BindingFlags) :
24 EventInfo[];

Description

When overridden in a derived class, searches for events that are declared or inherited by the current **System.Type**, using the specified binding constraints.

Return Value: An array of **System.Reflection.EventInfo** objects representing all events which are declared or inherited by the current **System.Type** that match the specified binding constraints.

The following **System.Reflection.BindingFlags** filter flags can be used to define which events to include in the search: *Public* to include public events in the search. A bitmask comprised of one or more **System.Reflection.BindingFlags** that specify how the search is conducted.

GetField

[C#] public FieldInfo GetField(string name);

[C++] public: FieldInfo* GetField(String* name);

[VB] Public Function GetField(ByVal name As String) As FieldInfo

[JScript] public function GetField(name : String) : FieldInfo;

Description

Searches for the field with the specified name.

Return Value: A **System.Reflection.FieldInfo** object representing the field with the specified name, if found; otherwise, **null**.

The search for *name* is case-sensitive. The **System.String** containing the name of the data field to get.

GetField

```

1
2 [C#] public abstract FieldInfo GetField(string name, BindingFlags bindingAttr);
3 [C++] public: virtual FieldInfo* GetField(String* name, BindingFlags
4 bindingAttr) = 0;
5 [VB] MustOverride Public Function GetField(ByVal name As String, ByVal
6 bindingAttr As BindingFlags) As FieldInfo
7 [JScript] public abstract function GetField(name : String, bindingAttr :
8 BindingFlags) : FieldInfo; Gets a specific field of the current System.Type .
9

```

Description

Searches for the specified field, using the specified binding constraints.

Return Value: A **System.Reflection.FieldInfo** object representing the field that matches the specified requirements, if found; otherwise, **null** .

The following table shows what members of a base class are returned by the **Get** methods when reflecting on a type. The **System.String** containing the name of the data field to get. A bitmask comprised of one or more **System.Reflection.BindingFlags** that specify how the search is conducted.

GetFields

```

18
19
20 [C#] public FieldInfo[] GetFields();
21 [C++] public: FieldInfo* GetFields() [];
22 [VB] Public Function GetFields() As FieldInfo()
23 [JScript] public function GetFields() : FieldInfo[]; Gets the fields of the current
24 System.Type .
25

```

Description

Returns all the public fields of the current **System.Type** .

Return Value: An array of **System.Reflection.FieldInfo** objects representing all the public fields defined for the current **System.Type** .

The following table shows what members of a base class are returned by the **Get** methods when reflecting on a type.

GetFields

[C#] public abstract FieldInfo[] GetFields(BindingFlags bindingAttr);

[C++] public: virtual FieldInfo* GetFields(BindingFlags bindingAttr) [] = 0;

[VB] MustOverride Public Function GetFields(ByVal bindingAttr As BindingFlags) As FieldInfo()

[JScript] public abstract function GetFields(bindingAttr : BindingFlags) : FieldInfo[];

Description

When overridden in a derived class, searches for the fields defined for the current **System.Type** , using the specified binding constraints.

Return Value: An array of **System.Reflection.FieldInfo** objects representing all fields defined for the current **System.Type** that match the specified binding constraints.

The following **System.Reflection.BindingFlags** filter flags can be used to define which fields to include in the search: *Instance* to include instance fields in

the search. A bitmask comprised of one or more **System.Reflection.BindingFlags** that specify how the search is conducted.

GetHashCode

[C#] public override int GetHashCode();

[C++] public: int GetHashCode();

[VB] Overrides Public Function GetHashCode() As Integer

[JScript] public override function GetHashCode() : int;

Description

Returns the hash code of the **System.Type** .

Return Value: The hash code of the **System.Type** .

GetInterface

[C#] public Type GetInterface(string name);

[C++] public: Type* GetInterface(String* name);

[VB] Public Function GetInterface(ByVal name As String) As Type

[JScript] public function GetInterface(name : String) : Type; Gets a specific interface implemented or inherited by the current **System.Type** .

Description

Searches for the interface with the specified name.

Return Value: A **System.Type** object representing the interface with the specified name, implemented or inherited by the current **System.Type** , if found; otherwise, **null** .

The search for *name* is case-sensitive. The **System.String** containing the name of the interface to get.

GetInterface

[C#] public abstract Type GetInterface(string name, bool ignoreCase);

[C++] public: virtual Type* GetInterface(String* name, bool ignoreCase) = 0;

[VB] MustOverride Public Function GetInterface(ByVal name As String, ByVal ignoreCase As Boolean) As Type

[JScript] public abstract function GetInterface(name : String, ignoreCase : Boolean) : Type;

Description

When overridden in a derived class, searches for the specified interface, specifying whether to do a case-sensitive search.

Return Value: A **System.Type** object representing the interface with the specified name, implemented or inherited by the current **System.Type**, if found; otherwise, **null**.

If *name* has 128 or more standard ASCII characters, a case-sensitive search is performed, regardless of the value of *ignoreCase*. Arrays or COM types are not searched for unless they have been previously loaded into the table of available classes. The **System.String** containing the name of the interface to get. **true** to perform a case-insensitive search for *name*.

GetInterfaceMap

[C#] public virtual InterfaceMapping GetInterfaceMap(Type interfaceType);

1 [C++] public: virtual InterfaceMapping GetInterfaceMap(Type* interfaceType);
 2 [VB] Overridable Public Function GetInterfaceMap(ByVal interfaceType As
 3 Type) As InterfaceMapping
 4 [JScript] public function GetInterfaceMap(interfaceType : Type) :
 5 InterfaceMapping;

7 *Description*

8 Returns an interface mapping for the specified interface type.

9 *Return Value:* An **System.Reflection.InterfaceMapping** object representing the
 10 interface mapping for *interfaceType* .

11 The interface map denotes how an interface is mapped into the actual
 12 methods on a class that implements that interface. The **System.Type** of the
 13 interface of which to retrieve a mapping.

14 **GetInterfaces**

15
 16 [C#] public abstract Type[] GetInterfaces();
 17 [C++] public: virtual Type* GetInterfaces() [] = 0;
 18 [VB] MustOverride Public Function GetInterfaces() As Type()
 19 [JScript] public abstract function GetInterfaces() : Type[];

21 *Description*

22 When overridden in a derived class, gets all the interfaces implemented or
 23 inherited by the current **System.Type** .

24 *Return Value:* An array of **System.Type** objects representing all the interfaces
 25 implemented or inherited by the current **System.Type** .

GetMember

```
[C#] public MemberInfo[] GetMember(string name);  
[C++] public: MemberInfo* GetMember(String* name) [];  
[VB] Public Function GetMember(ByVal name As String) As MemberInfo()  
[JScript] public function GetMember(name : String) : MemberInfo[]; Gets the  
specified members of the current System.Type .
```

Description

Searches for the members with the specified name.

Return Value: An array of **System.Reflection.MemberInfo** objects representing the public members with the specified name, if found; otherwise, **null** .

The search for *name* is case-sensitive. The **System.String** containing the name of the public members to get.

GetMember

```
[C#] public virtual MemberInfo[] GetMember(string name, BindingFlags  
bindingAttr);  
[C++] public: virtual MemberInfo* GetMember(String* name, BindingFlags  
bindingAttr) [];  
[VB] Overridable Public Function GetMember(ByVal name As String, ByVal  
bindingAttr As BindingFlags) As MemberInfo()  
[JScript] public function GetMember(name : String, bindingAttr : BindingFlags) :  
MemberInfo[];
```

Description

Searches for the specified members, using the specified binding constraints.

Return Value: An array of **System.Reflection.MemberInfo** objects representing the public members with the specified name, if found; otherwise, **null**.

This method can be overridden by a derived class. The **System.String** containing the name of the members to get. A bitmask comprised of one or more **System.Reflection.BindingFlags** that specify how the search is conducted.

GetMember

[C#] public virtual MemberInfo[] GetMember(string name, MemberTypes type, BindingFlags bindingAttr);

[C++] public: virtual MemberInfo* GetMember(String* name, MemberTypes type, BindingFlags bindingAttr) [];

[VB] Overridable Public Function GetMember(ByVal name As String, ByVal type As MemberTypes, ByVal bindingAttr As BindingFlags) As MemberInfo()

[JScript] public function GetMember(name : String, type : MemberTypes, bindingAttr : BindingFlags) : MemberInfo[];

Description

Searches for the specified members of the specified member type, using the specified binding constraints.

Return Value: An array of **System.Reflection.MemberInfo** objects representing the public members with the specified name, if found; otherwise, **null**.

Members include properties, methods, fields, events, and so on. The **System.String** containing the name of the members to get. The **System.Type** of member to search for. A bitmask comprised of one or more **System.Reflection.BindingFlags** that specify how the search is conducted.

GetMembers

[C#] public MemberInfo[] GetMembers();

[C++] public: MemberInfo* GetMembers() [];

[VB] Public Function GetMembers() As MemberInfo()

[JScript] public function GetMembers() : MemberInfo[]; Gets the members (properties, methods, fields, events, and so on) of the current **System.Type**.

Description

Returns all the public members of the current **System.Type**.

Return Value: An array of **System.Reflection.MemberInfo** objects representing all the public members of the current **System.Type**.

Members include properties, methods, fields, events, and so on.

GetMembers

[C#] public abstract MemberInfo[] GetMembers(BindingFlags bindingAttr);

[C++] public: virtual MemberInfo* GetMembers(BindingFlags bindingAttr) [] = 0;

[VB] MustOverride Public Function GetMembers(ByVal bindingAttr As BindingFlags) As MemberInfo()

[JScript] public abstract function GetMembers(bindingAttr : BindingFlags) :

1 MemberInfo[];

2
3 *Description*

4 When overridden in a derived class, searches for the members defined for
5 the current **System.Type** , using the specified binding constraints.

6 *Return Value:* An array of **System.Reflection.MemberInfo** objects representing
7 all members defined for the current **System.Type** that match the specified binding
8 constraints.

9 Members include properties, methods, fields, events, and so on. A bitmask
10 comprised of one or more **System.Reflection.BindingFlags** that specify how the
11 search is conducted.

12 **GetMethod**

13
14 [C#] public MethodInfo GetMethod(string name);

15 [C++] public: MethodInfo* GetMethod(String* name);

16 [VB] Public Function GetMethod(ByVal name As String) As MethodInfo

17 [JScript] public function GetMethod(name : String) : MethodInfo;

18
19 *Description*

20 Searches for the public method with the specified name.

21 *Return Value:* A **System.Reflection.MethodInfo** object representing the public
22 method with the specified name, if found; otherwise, **null** .

23 The search for *name* is case-sensitive. The **System.String** containing the
24 name of the public method to get.

25 **GetMethod**

```

1
2 [C#] public MethodInfo GetMethod(string name, BindingFlags bindingAttr);
3 [C++] public: __sealed MethodInfo* GetMethod(String* name, BindingFlags
4 bindingAttr);
5 [VB] NotOverridable Public Function GetMethod(ByVal name As String, ByVal
6 bindingAttr As BindingFlags) As MethodInfo
7 [JScript] public function GetMethod(name : String, bindingAttr : BindingFlags) :
8 MethodInfo;
9

```

Description

Searches for the specified method, using the specified binding constraints.

Return Value: A **System.Reflection.MethodInfo** object representing the method that matches the specified requirements, if found; otherwise, **null**.

The following **System.Reflection.BindingFlags** filter flags can be used to define which methods to include in the search: *Instance* to include instance methods in the search. The **System.String** containing the name of the method to get. A bitmask comprised of one or more **System.Reflection.BindingFlags** that specify how the search is conducted.

GetMethod

```

20
21 [C#] public MethodInfo GetMethod(string name, Type[] types);
22 [C++] public: MethodInfo* GetMethod(String* name, Type* types[]);
23 [VB] Public Function GetMethod(ByVal name As String, ByVal types() As Type)
24 As MethodInfo
25 [JScript] public function GetMethod(name : String, types : Type[]) : MethodInfo;

```

Description

Searches for the specified public method whose parameters match the specified argument types.

Return Value: A **System.Reflection.MethodInfo** object representing the public method whose parameters match the specified argument types, if found; otherwise, **null**.

The search for *name* is case-sensitive. The **System.String** containing the name of the public method to get. An array of **System.Type** objects representing the number, order, and type of the parameters for the method to get.

GetMethod

[C#] public MethodInfo GetMethod(string name, Type[] types, ParameterModifier[] modifiers);

[C++] public: MethodInfo* GetMethod(String* name, Type* types[], ParameterModifier modifiers[]);

[VB] Public Function GetMethod(ByVal name As String, ByVal types() As Type, ByVal modifiers() As ParameterModifier) As MethodInfo

[JScript] public function GetMethod(name : String, types : Type[], modifiers : ParameterModifier[]) : MethodInfo;

Description

Searches for the specified public method whose parameters match the specified argument types and modifiers.

1 *Return Value:* A **System.Reflection.MethodInfo** object representing the public
2 method that matches the specified requirements, if found; otherwise, **null** .

3 The *types* array and the *modifiers* array have the same length. A parameter
4 specified in the *types* array can have the following attributes, which are specified
5 in the *modifiers* array: *pdIn*, *pdOut*, *pdLcid*, *pdRetval*, *pdOptional*, and
6 *pdHasDefault*, which represent [In], [Out], [lcid], [retval], [optional], and a value
7 specifying whether the parameter has a default value. A parameter's associated
8 attributes are stored in the metadata and are used for interoperability. The
9 **System.String** containing the name of the public method to get. An array of
10 **System.Type** objects representing the number, order, and type of the parameters
11 for the method to get. An array of **System.Reflection.ParameterModifier** objects
12 representing the attributes associated with the corresponding element in the *types*
13 array.

14 GetMethod

15
16 [C#] public MethodInfo GetMethod(string name, BindingFlags bindingAttr,
17 Binder binder, Type[] types, ParameterModifier[] modifiers);
18 [C++] public: __sealed MethodInfo* GetMethod(String* name, BindingFlags
19 bindingAttr, Binder* binder, Type* types[], ParameterModifier modifiers[]);
20 [VB] NotOverridable Public Function GetMethod(ByVal name As String, ByVal
21 bindingAttr As BindingFlags, ByVal binder As Binder, ByVal types() As Type,
22 ByVal modifiers() As ParameterModifier) As MethodInfo
23 [JScript] public function GetMethod(name : String, bindingAttr : BindingFlags,
24 binder : Binder, types : Type[], modifiers : ParameterModifier[]) : MethodInfo;
25

Description

Searches for the specified method whose parameters match the specified argument types and modifiers, using the specified binding constraints.

Return Value: A **System.Reflection.MethodInfo** object representing the method that matches the specified requirements, if found; otherwise, **null**.

The *types* array and the *modifiers* array have the same length. A parameter specified in the *types* array can have the following attributes, which are specified in the *modifiers* array: *pdIn*, *pdOut*, *pdLcid*, *pdRetVal*, *pdOptional*, and *pdHasDefault*, which represent [In], [Out], [lcid], [retval], [optional], and a value specifying whether the parameter has a default value. A parameter's associated attributes are stored in the metadata and enhance interoperability. The **System.String** containing the name of the method to get. A bitmask comprised of one or more **System.Reflection.BindingFlags** that specify how the search is conducted. A **System.Reflection.Binder** object that defines a set of properties and enables binding, which can involve selection of an overloaded method, coercion of argument types, and invocation of a member through reflection. An array of **System.Type** objects representing the number, order, and type of the parameters for the method to get. An array of **System.Reflection.ParameterModifier** objects representing the attributes associated with the corresponding element in the *types* array.

GetMethod

```
[C#] public MethodInfo GetMethod(string name, BindingFlags bindingAttr,  
Binder binder, CallingConventions callConvention, Type[] types,
```

```

1 ParameterModifier[] modifiers);
2 [C++] public: MethodInfo* GetMethod(String* name, BindingFlags bindingAttr,
3 Binder* binder, CallingConventions callConvention, Type* types[],
4 ParameterModifier modifiers[]);
5 [VB] Public Function GetMethod(ByVal name As String, ByVal bindingAttr As
6 BindingFlags, ByVal binder As Binder, ByVal callConvention As
7 CallingConventions, ByVal types() As Type, ByVal modifiers() As
8 ParameterModifier) As MethodInfo
9 [JScript] public function GetMethod(name : String, bindingAttr : BindingFlags,
10 binder : Binder, callConvention : CallingConventions, types : Type[], modifiers :
11 ParameterModifier[]) : MethodInfo; Gets a specific method of the current
12 System.Type .

```

Description

Searches for the specified method whose parameters match the specified argument types and modifiers, using the specified binding constraints and the specified calling convention.

Return Value: A **System.Reflection.MethodInfo** object representing the method that matches the specified requirements, if found; otherwise, **null** .

The following table shows what members of a base class are returned by the **GetXXX** methods when reflecting on a type. The **System.String** containing the name of the method to get. A bitmask comprised of one or more **System.Reflection.BindingFlags** that specify how the search is conducted. A **System.Reflection.Binder** object that defines a set of properties and enables binding, which can involve selection of an overloaded method, coercion of

argument types, and invocation of a member through reflection. The **System.Reflection.CallingConventions** object that specifies the set of rules to use regarding the order and layout of arguments, how the return value is passed, what registers are used for arguments, and how the stack is cleaned up. An array of **System.Type** objects representing the number, order, and type of the parameters for the method to get. An array of **System.Reflection.ParameterModifier** objects representing the attributes associated with the corresponding element in the *types* array.

GetMethodImpl

[C#] protected abstract MethodInfo GetMethodImpl(string name, BindingFlags bindingAttr, Binder binder, CallingConventions callConvention, Type[] types, ParameterModifier[] modifiers);

[C++] protected: virtual MethodInfo* GetMethodImpl(String* name, BindingFlags bindingAttr, Binder* binder, CallingConventions callConvention, Type* types[], ParameterModifier modifiers[]) = 0;

[VB] MustOverride Protected Function GetMethodImpl(ByVal name As String, ByVal bindingAttr As BindingFlags, ByVal binder As Binder, ByVal callConvention As CallingConventions, ByVal types() As Type, ByVal modifiers() As ParameterModifier) As MethodInfo

[JScript] protected abstract function GetMethodImpl(name : String, bindingAttr : BindingFlags, binder : Binder, callConvention : CallingConventions, types : Type[], modifiers : ParameterModifier[]) : MethodInfo;

Description

When overridden in a derived class, searches for the specified method whose parameters match the specified argument types and modifiers, using the specified binding constraints and the specified calling convention.

Return Value: A **System.Reflection.MethodInfo** object representing the method that matches the specified requirements, if found; otherwise, **null**.

If *types* is **null**, arguments are not matched. The **System.String** containing the name of the method to get. A bitmask comprised of one or more **System.Reflection.BindingFlags** that specify how the search is conducted. A **System.Reflection.Binder** object that defines a set of properties and enables binding, which can involve selection of an overloaded method, coercion of argument types, and invocation of a member through reflection. The **System.Reflection.CallingConventions** object that specifies the set of rules to use regarding the order and layout of arguments, how the return value is passed, what registers are used for arguments, and what process cleans up the stack. An array of **System.Type** objects representing the number, order, and type of the parameters for the method to get. An array of **System.Reflection.ParameterModifier** objects representing the attributes associated with the corresponding element in the *types* array.

GetMethods

[C#] public MethodInfo[] GetMethods();

[C++] public: MethodInfo* GetMethods() [];

[VB] Public Function GetMethods() As MethodInfo()

[JScript] public function GetMethods() : MethodInfo[]; Gets the methods of the current **System.Type**.

Description

Returns all the public methods of the current **System.Type** .

Return Value: An array of **System.Reflection.MethodInfo** objects representing all the public methods defined for the current **System.Type** .

The following table shows what members of a base class are returned by the **Get** methods when reflecting on a type.

GetMethods

[C#] public abstract MethodInfo[] GetMethods(BindingFlags bindingAttr);

[C++] public: virtual MethodInfo* GetMethods(BindingFlags bindingAttr) [] = 0;

[VB] MustOverride Public Function GetMethods(ByVal bindingAttr As BindingFlags) As MethodInfo()

[JScript] public abstract function GetMethods(bindingAttr : BindingFlags) : MethodInfo[];

Description

When overridden in a derived class, searches for the methods defined for the current **System.Type** , using the specified binding constraints.

Return Value: An array of **System.Reflection.MethodInfo** objects representing all methods defined for the current **System.Type** that match the specified binding constraints.

The following **System.Reflection.BindingFlags** filter flags can be used to define which methods to include in the search: *Instance* to include instance

1 methods in the search. A bitmask comprised of one or more

2 **System.Reflection.BindingFlags** that specify how the search is conducted.

3 GetNestedType

4
5 [C#] public Type GetNestedType(string name);

6 [C++] public: Type* GetNestedType(String* name);

7 [VB] Public Function GetNestedType(ByVal name As String) As Type

8 [JScript] public function GetNestedType(name : String) : Type; Gets a specific
9 type nested within the current **System.Type** .

10
11 *Description*

12 Searches for the nested type with the specified name.

13 *Return Value:* A **System.Type** object representing the nested type with the
14 specified name, if found; otherwise, **null** .

15 The search for *name* is case-sensitive. The **System.String** containing the
16 name of the nested type to get.

17 GetNestedType

18
19 [C#] public abstract Type GetNestedType(string name, BindingFlags bindingAttr);

20 [C++] public: virtual Type* GetNestedType(String* name, BindingFlags
21 bindingAttr) = 0;

22 [VB] MustOverride Public Function GetNestedType(ByVal name As String,
23 ByVal bindingAttr As BindingFlags) As Type

24 [JScript] public abstract function GetNestedType(name : String, bindingAttr :
25 BindingFlags) : Type;

Description

When overridden in a derived class, searches for the specified nested type, using the specified binding constraints.

Return Value: A **System.Type** object representing the nested type that matches the specified requirements, if found; otherwise, **null**.

The following **System.Reflection.BindingFlags** filter flags can be used to define which nested types to include in the search: *Public* to include public nested types in the search. The **System.String** containing the name of the nested type to get. A bitmask comprised of one or more **System.Reflection.BindingFlags** that specify how the search is conducted.

GetNestedTypes

[C#] public Type[] GetNestedTypes();

[C++] public: Type* GetNestedTypes() [];

[VB] Public Function GetNestedTypes() As Type()

[JScript] public function GetNestedTypes() : Type[]; Gets the types nested within the current **System.Type**.

Description

Returns all the types nested within the current **System.Type**.

Return Value: An array of **System.Type** objects representing all the types nested within the current **System.Type**.

The following table shows what members of a base class are returned by the **Get** methods when reflecting on a type.

GetNestedTypes

```
[C#] public abstract Type[] GetNestedTypes(BindingFlags bindingAttr);  
[C++] public: virtual Type* GetNestedTypes(BindingFlags bindingAttr) [] = 0;  
[VB] MustOverride Public Function GetNestedTypes(ByVal bindingAttr As  
BindingFlags) As Type()  
[JScript] public abstract function GetNestedTypes(bindingAttr : BindingFlags) :  
Type[];
```

Description

When overridden in a derived class, searches for the types nested within the current **System.Type**, using the specified binding constraints.

Return Value: An array of **System.Type** objects representing all the types nested within the current **System.Type** that match the specified binding constraints.

The following **System.Reflection.BindingFlags** filter flags can be used to define which nested types to include in the search: *Public* to include public nested types in the search. A bitmask comprised of one or more **System.Reflection.BindingFlags** that specify how the search is conducted.

GetProperties

```
[C#] public PropertyInfo[] GetProperties();  
[C++] public: PropertyInfo* GetProperties() [];  
[VB] Public Function GetProperties() As PropertyInfo()  
[JScript] public function GetProperties() : PropertyInfo[];
```

Description

Returns all the public properties of the current **System.Type** .

Return Value: An array of **System.Reflection.PropertyInfo** objects representing all public properties of the current **System.Type** .

The following table shows what members of a base class are returned by the **Get** methods when reflecting on a type.

GetProperties

```
[C#] public abstract PropertyInfo[] GetProperties(BindingFlags bindingAttr);  
[C++] public: virtual PropertyInfo* GetProperties(BindingFlags bindingAttr) [] =  
0;  
[VB] MustOverride Public Function GetProperties(ByVal bindingAttr As  
BindingFlags) As PropertyInfo()  
[JScript] public abstract function GetProperties(bindingAttr : BindingFlags) :  
PropertyInfo[]; Gets the properties of the current System.Type .
```

Description

When overridden in a derived class, searches for the properties of the current **System.Type** , using the specified binding constraints.

Return Value: An array of **System.Reflection.PropertyInfo** objects representing all properties of the current **System.Type** that match the specified binding constraints.

A property is considered public to reflection if it has at least one accessor that is public. That is, you can call type.GetProperty("propertyname"),

BindingFlags.Public | BindingFlags.Instance | BindingFlags.Static) to get it. A bitmask comprised of one or more **System.Reflection.BindingFlags** that specify how the search is conducted.

GetProperty

[C#] public PropertyInfo GetProperty(string name);

[C++] public: PropertyInfo* GetProperty(String* name);

[VB] Public Function GetProperty(ByVal name As String) As PropertyInfo

[JScript] public function GetProperty(name : String) : PropertyInfo;

Description

Searches for the public property with the specified name.

Return Value: A **System.Reflection.PropertyInfo** object representing the public property with the specified name, if found; otherwise, **null**.

The search for *name* is case-sensitive. The **System.String** containing the name of the public property to get.

GetProperty

[C#] public PropertyInfo GetProperty(string name, BindingFlags bindingAttr);

[C++] public: __sealed PropertyInfo* GetProperty(String* name, BindingFlags bindingAttr);

[VB] NotOverridable Public Function GetProperty(ByVal name As String, ByVal bindingAttr As BindingFlags) As PropertyInfo

[JScript] public function GetProperty(name : String, bindingAttr : BindingFlags) : PropertyInfo;

Description

Searches for the specified property, using the specified binding constraints.

Return Value: A **System.Reflection.PropertyInfo** object representing the property that matches the specified requirements, if found; otherwise, **null**.

The *types* array and the *modifiers* array have the same length. A parameter specified in the *types* array can have the following attributes, which are specified in the *modifiers* array: *pdIn*, *pdOut*, *pdLcid*, *pdRetVal*, *pdOptional*, and *pdHasDefault*, which represent [In], [Out], [lcid], [retval], [optional], and a value specifying whether the parameter has a default value. A parameter's associated attributes are stored in the metadata and enhance interoperability. The **System.String** containing the name of the property to get. A bitmask comprised of one or more **System.Reflection.BindingFlags** that specify how the search is conducted.

GetProperty

[C#] public PropertyInfo GetProperty(string name, Type returnType);

[C++] public: PropertyInfo* GetProperty(String* name, Type* returnType);

[VB] Public Function GetProperty(ByVal name As String, ByVal returnType As Type) As PropertyInfo

[JScript] public function GetProperty(name : String, returnType : Type) :

PropertyInfo;

Description

Searches for the public property with the specified name and return type.

Return Value: A **System.Reflection.PropertyInfo** object representing the public property with the specified name, if found; otherwise, **null**.

The search for *name* is case-sensitive. The **System.String** containing the name of the public property to get. The return type of the property.

GetProperty

[C#] public PropertyInfo GetProperty(string name, Type[] types);

[C++] public: PropertyInfo* GetProperty(String* name, Type* types[]);

[VB] Public Function GetProperty(ByVal name As String, ByVal types() As Type) As PropertyInfo

[JScript] public function GetProperty(name : String, types : Type[]) :

PropertyInfo;

Description

Searches for the specified public property whose parameters match the specified argument types.

Return Value: A **System.Reflection.PropertyInfo** object representing the public property whose parameters match the specified argument types, if found; otherwise, **null**.

The search for *name* is case-sensitive. The **System.String** containing the name of the public property to get. An array of **System.Type** objects representing the number, order, and type of the parameters for the indexed property to get.

GetProperty

```

1
2 [C#] public PropertyInfo GetProperty(string name, Type returnType, Type[]
3 types);
4 [C++] public: PropertyInfo* GetProperty(String* name, Type* returnType, Type*
5 types[]);
6 [VB] Public Function GetProperty(ByVal name As String, ByVal returnType As
7 Type, ByVal types() As Type) As PropertyInfo
8 [JScript] public function GetProperty(name : String, returnType : Type, types :
9 Type[]) : PropertyInfo;
10

```

Description

Searches for the specified public property whose parameters match the specified argument types.

Return Value: A **System.Reflection.PropertyInfo** object representing the public property whose parameters match the specified argument types, if found; otherwise, **null**.

The search for *name* is case-sensitive. The **System.String** containing the name of the public property to get. The return type of the property. An array of **System.Type** objects representing the number, order, and type of the parameters for the indexed property to get.

GetProperty

```

23 [C#] public PropertyInfo GetProperty(string name, Type returnType, Type[] types,
24 ParameterModifier[] modifiers);
25 [C++] public: PropertyInfo* GetProperty(String* name, Type* returnType, Type*

```

1 types[], ParameterModifier modifiers[]);

2 [VB] Public Function GetProperty(ByVal name As String, ByVal returnType As
3 Type, ByVal types() As Type, ByVal modifiers() As ParameterModifier) As
4 PropertyInfo

5 [JScript] public function GetProperty(name : String, returnType : Type, types :
6 Type[], modifiers : ParameterModifier[]) : PropertyInfo;

8 *Description*

9 Searches for the specified public property whose parameters match the
10 specified argument types and modifiers.

11 *Return Value:* A **System.Reflection.PropertyInfo** object representing the public
12 property that matches the specified requirements, if found; otherwise, **null** .

13 The *types* array and the *modifiers* array have the same length. A parameter
14 specified in the *types* array can have the following attributes, which are specified
15 in the *modifiers* array: *pdIn*, *pdOut*, *pdLcid*, *pdRetVal*, *pdOptional*, and
16 *pdHasDefault*, which represent [In], [Out], [lcid], [retval], [optional], and a value
17 specifying whether the parameter has a default value. A parameter's associated
18 attributes are stored in the metadata and enhance interoperability. The
19 **System.String** containing the name of the public property to get. The return type
20 of the property. An array of **System.Type** objects representing the number, order,
21 and type of the parameters for the indexed property to get. An array of
22 **System.Reflection.ParameterModifier** objects representing the attributes
23 associated with the corresponding element in the *types* array.

24 **GetProperty**

```
[C#] public PropertyInfo GetProperty(string name, BindingFlags bindingAttr,
Binder binder, Type returnType, Type[] types, ParameterModifier[] modifiers);

[C++] public: __sealed PropertyInfo* GetProperty(String* name, BindingFlags
bindingAttr, Binder* binder, Type* returnType, Type* types[], ParameterModifier
modifiers[]);

[VB] NotOverridable Public Function GetProperty(ByVal name As String, ByVal
bindingAttr As BindingFlags, ByVal binder As Binder, ByVal returnType As
Type, ByVal types() As Type, ByVal modifiers() As ParameterModifier) As
PropertyInfo

[JScript] public function GetProperty(name : String, bindingAttr : BindingFlags,
binder : Binder, returnType : Type, types : Type[], modifiers :
ParameterModifier[]) : PropertyInfo; Gets a specific property of the current
System.Type .
```

Description

Searches for the specified property whose parameters match the specified argument types and modifiers, using the specified binding constraints.

Return Value: A **System.Reflection.PropertyInfo** object representing the property that matches the specified requirements, if found; otherwise, **null**.

The following table shows what members of a base class are returned by the **Get** methods when reflecting on a type. The **System.String** containing the name of the property to get. A bitmask comprised of one or more **System.Reflection.BindingFlags** that specify how the search is conducted. A **System.Reflection.Binder** object that defines a set of properties and enables

binding, which can involve selection of an overloaded method, coercion of argument types, and invocation of a member through reflection. The return type of the property. An array of **System.Type** objects representing the number, order, and type of the parameters for the indexed property to get. An array of **System.Reflection.ParameterModifier** objects representing the attributes associated with the corresponding element in the *types* array.

GetPropertyImpl

[C#] protected abstract PropertyInfo GetPropertyImpl(string name, BindingFlags bindingAttr, Binder binder, Type returnType, Type[] types, ParameterModifier[] modifiers);

[C++] protected: virtual PropertyInfo* GetPropertyImpl(String* name, BindingFlags bindingAttr, Binder* binder, Type* returnType, Type* types[], ParameterModifier modifiers[]) = 0;

[VB] MustOverride Protected Function GetPropertyImpl(ByVal name As String, ByVal bindingAttr As BindingFlags, ByVal binder As Binder, ByVal returnType As Type, ByVal types() As Type, ByVal modifiers() As ParameterModifier) As PropertyInfo

[JScript] protected abstract function GetPropertyImpl(name : String, bindingAttr : BindingFlags, binder : Binder, returnType : Type, types : Type[], modifiers : ParameterModifier[]) : PropertyInfo;

Description

When overridden in a derived class, searches for the specified property whose parameters match the specified argument types and modifiers, using the

specified binding constraints.

Return Value: A **System.Reflection.PropertyInfo** object representing the property that matches the specified requirements, if found; otherwise, **null**.

The *types* array and the *modifiers* array have the same length. A parameter specified in the *types* array can have the following attributes, which are specified in the *modifiers* array: **pdIn**, **pdOut**, **pdLcid**, **pdRetVal**, **pdOptional**, and **pdHasDefault**, which represent [In], [Out], [lcid], [retval], [optional], and a value specifying whether the parameter has a default value. A parameter's associated attributes are stored in the metadata and are used for interoperability. The **System.String** containing the name of the property to get. A bitmask comprised of one or more **System.Reflection.BindingFlags** that specify how the search is conducted. A **System.Reflection.Binder** object that defines a set of properties and enables binding, which can involve selection of an overloaded member, coercion of argument types, and invocation of a member through reflection. The return type of the property. An array of **System.Type** objects representing the number, order, and type of the parameters for the indexed property to get. An array of **System.Reflection.ParameterModifier** objects representing the attributes associated with the corresponding element in the *types* array.

GetType

[C#] public static Type GetType(string typeName);

[C++] public: static Type* GetType(String* typeName);

[VB] Public Shared Function GetType(ByVal typeName As String) As Type

[JScript] public static function GetType(typeName : String) : Type;

Description

Gets the **System.Type** with the specified name, performing a case-sensitive search.

Return Value: The **System.Type** with the specified name, if found; otherwise, **null**

GetType only works on assemblies loaded from disk. If you call **GetType** to look up a type defined in a dynamic assembly defined using the **System.Reflection.Emit** services, you might get inconsistent behavior. The behavior depends on whether the dynamic assembly is persistent, that is, created using the **RunAndSave** or **Save** access modes of the **System.Reflection.Emit.AssemblyBuilderAccess** enumeration. If the dynamic assembly is persistent and has been written to disk before **GetType** is called, the loader finds the saved assembly on disk, loads that assembly, and retrieves the type from that assembly. If the assembly has not been saved to disk when **GetType** is called, the method returns **null**. The name of the **System.Type** to get.

GetType

[C#] public static Type GetType(string typeName, bool throwOnError);

[C++] public: static Type* GetType(String* typeName, bool throwOnError);

[VB] Public Shared Function GetType(ByVal typeName As String, ByVal throwOnError As Boolean) As Type

[JScript] public static function GetType(typeName : String, throwOnError : Boolean) : Type;

Description

Gets the **System.Type** with the specified name, performing a case-sensitive search and specifying whether to throw an exception if an error occurs while loading the **System.Type**.

Return Value: The **System.Type** with the specified name, if found; otherwise, **null**.

GetType only works on assemblies loaded from disk. If you call **GetType** to look up a type defined in a dynamic assembly defined using the **System.Reflection.Emit** services, you might get inconsistent behavior. The behavior depends on whether the dynamic assembly is persistent, that is, created using the **RunAndSave** or **Save** access modes of the **System.Reflection.Emit.AssemblyBuilderAccess** enumeration. If the dynamic assembly is persistent and has been written to disk before **GetType** is called, the loader finds the saved assembly on disk, loads that assembly, and retrieves the type from that assembly. If the assembly has not been saved to disk when **GetType** is called, the method returns **null**. The name of the **System.Type** to get. **true** to throw a **System.TypeLoadException** if an error occurs while loading the **System.Type**.

GetType

```
[C#] public static Type GetType(string typeName, bool throwOnError, bool ignoreCase);
```

```
[C++] public: static Type* GetType(String* typeName, bool throwOnError, bool ignoreCase);
```


[VB] Public Shared Function GetType(ByVal typeName As String, ByVal
throwOnError As Boolean, ByVal ignoreCase As Boolean) As Type
[JScript] public static function GetType(typeName : String, throwOnError :
Boolean, ignoreCase : Boolean) : Type; Gets the **System.Type** with the specified
name.

Description

Gets the **System.Type** with the specified name, specifying whether to
perform a case-sensitive search and whether to throw an exception if an error
occurs while loading the **System.Type**.

Return Value: The **System.Type** with the specified name, if found; otherwise, **null**

GetType only works on assemblies loaded from disk. If you call **GetType**
to look up a type defined in a dynamic assembly defined using the
System.Reflection.Emit services, you might get inconsistent behavior. The
behavior depends on whether the dynamic assembly is persistent, that is, created
using the **RunAndSave** or **Save** access modes of the
System.Reflection.Emit.AssemblyBuilderAccess enumeration. If the dynamic
assembly is persistent and has been written to disk before **GetType** is called, the
loader finds the saved assembly on disk, loads that assembly, and retrieves the
type from that assembly. If the assembly has not been saved to disk when
GetType is called, the method returns **null**. The name of the **System.Type** to get.
true to throw a **System.TypeLoadException** if an error occurs while loading the
System.Type. **true** to perform a case-insensitive search for *typeName*, if
typeName has less than 128 characters.

GetTypeInfo

```
[C#] public static Type[] GetTypeInfo(object[] args);  
[C++] public: static Type* GetTypeInfo(Object* args __gc[]) [];  
[VB] Public Shared Function GetTypeInfo(ByVal args() As Object) As Type()  
[JScript] public static function GetTypeInfo(args : Object[]) : Type[];
```

Description

Gets the types of the objects in the specified array.

Return Value: An array of **System.Type** objects representing the types of the corresponding elements in *args* . An array of objects whose types to determine.

GetTypeCode

```
[C#] public static TypeCode GetTypeCode(Type type);  
[C++] public: static TypeCode GetTypeCode(Type* type);  
[VB] Public Shared Function GetTypeCode(ByVal type As Type) As TypeCode  
[JScript] public static function GetTypeCode(type : Type) : TypeCode;
```

Description

Gets the underlying type code of the specified **System.Type** .

Return Value: The **System.TypeCode** value of the underlying type. The **System.Type** whose underlying type code to get.

GetTypeFromCLSID

```
[C#] public static Type GetTypeFromCLSID(Guid clsid);
```

1 [C++] public: static Type* GetTypeFromCLSID(Guid clsid);

2 [VB] Public Shared Function GetTypeFromCLSID(ByVal clsid As Guid) As Type

3 [JScript] public static function GetTypeFromCLSID(clsid : Guid) : Type; Gets the

4 **System.Type** associated with the specified class identifier (CLSID).

5
6 *Description*

7 Gets the **System.Type** associated with the specified class identifier
8 (CLSID).

9 *Return Value:* **System.__ComObject** regardless of whether the CLSID is valid.

10 The CLSID of the **System.Type** to get.

11 **GetTypeFromCLSID**

12
13 [C#] public static Type GetTypeFromCLSID(Guid clsid, bool throwOnError);

14 [C++] public: static Type* GetTypeFromCLSID(Guid clsid, bool throwOnError);

15 [VB] Public Shared Function GetTypeFromCLSID(ByVal clsid As Guid, ByVal
16 throwOnError As Boolean) As Type

17 [JScript] public static function GetTypeFromCLSID(clsid : Guid, throwOnError :
18 Boolean) : Type;

19
20 *Description*

21 Gets the **System.Type** associated with the specified class identifier
22 (CLSID), specifying whether to throw an exception if an error occurs while
23 loading the **System.Type** .

24 *Return Value:* **System.__ComObject** regardless of whether the CLSID is valid.

Exceptions such as **System.OutOfMemoryException** will be thrown when specifying **true** for *throwOnError* , but it will not fail for unregistered CLSID's.

The CLSID of the **System.Type** to get. **true** to throw a **System.TypeLoadException** if an error occurs while loading the **System.Type**.

GetTypeFromCLSID

[C#] public static Type GetTypeFromCLSID(Guid clsid, string server);

[C++] public: static Type* GetTypeFromCLSID(Guid clsid, String* server);

[VB] Public Shared Function GetTypeFromCLSID(ByVal clsid As Guid, ByVal server As String) As Type

[JScript] public static function GetTypeFromCLSID(clsid : Guid, server : String) : Type;

Description

Gets the **System.Type** associated with the specified class identifier (CLSID) from the specified server.

Return Value: **System.__ComObject** regardless of whether the CLSID is valid. The CLSID of the **System.Type** to get. The server from which to load the type.

GetTypeFromCLSID

[C#] public static Type GetTypeFromCLSID(Guid clsid, string server, bool throwOnError);

[C++] public: static Type* GetTypeFromCLSID(Guid clsid, String* server, bool throwOnError);

[VB] Public Shared Function GetTypeFromCLSID(ByVal clsid As Guid, ByVal

server As String, ByVal throwOnError As Boolean) As Type

[JScript] public static function GetTypeFromCLSID(clsid : Guid, server : String,
throwOnError : Boolean) : Type;

Description

Gets the **System.Type** associated with the specified class identifier (CLSID) from the specified server, specifying whether to throw an exception if an error occurs while loading the **System.Type**.

Return Value: **System.__ComObject** regardless of whether the CLSID is valid.

Exceptions such as **System.OutOfMemoryException** will be thrown when specifying **true** for *throwOnError*, but it will not fail for unregistered CLSID's.

The CLSID of the **System.Type** to get. The server from which to load the type.

true to throw a **System.TypeLoadException** if an error occurs while loading the **System.Type**.

GetTypeFromHandle

[C#] public static Type GetTypeFromHandle(RuntimeTypeHandle handle);

[C++] public: static Type* GetTypeFromHandle(RuntimeTypeHandle handle);

[VB] Public Shared Function GetTypeFromHandle(ByVal handle As
RuntimeTypeHandle) As Type

[JScript] public static function GetTypeFromHandle(handle :
RuntimeTypeHandle) : Type;

Description

Gets the **System.Type** referenced by the specified type handle.

Return Value: The **System.Type** referenced by the specified

System.RuntimeTypeHandle .

The handles are valid only in the application domain in which they were obtained. The **System.RuntimeTypeHandle** object that refers to the **System.Type**.

GetTypeFromProgID

[C#] public static Type GetTypeFromProgID(string progID);

[C++] public: static Type* GetTypeFromProgID(String* progID);

[VB] Public Shared Function GetTypeFromProgID(ByVal progID As String) As Type

[JScript] public static function GetTypeFromProgID(progID : String) : Type; Gets the **System.Type** associated with the specified program identifier (PROGID).

Description

Gets the **System.Type** associated with the specified program identifier (PROGID), returning null if an error is encountered while loading the **System.Type** .

Return Value: The **System.Type** associated with the specified PROGID, if *progID* is a valid entry in the registry and a type is associated with it; otherwise, **null** .

This method is provided for COM support. PROGIDs are not used in the Microsoft.NET Framework because they have been superseded by the concept of namespace. The PROGID of the **System.Type** to get.

GetTypeFromProgID

```

1
2 [C#] public static Type GetTypeFromProgID(string progID, bool throwOnError);
3 [C++] public: static Type* GetTypeFromProgID(String* progID, bool
4 throwOnError);
5 [VB] Public Shared Function GetTypeFromProgID(ByVal progID As String,
6 ByVal throwOnError As Boolean) As Type
7 [JScript] public static function GetTypeFromProgID(progID : String,
8 throwOnError : Boolean) : Type;
9

```

Description

Gets the **System.Type** associated with the specified program identifier (PROGID), specifying whether to throw an exception if an error occurs while loading the **System.Type**.

Return Value: The **System.Type** associated with the specified program identifier (PROGID), if *progID* is a valid entry in the registry and a type is associated with it; otherwise, **null**.

This method is provided for COM support. Program IDs are not used in Microsoft .NET Framework because they have been superseded by the concept of namespace. The PROGID of the **System.Type** to get. **true** to throw a **System.TypeLoadException** if an error occurs while loading the **System.Type**.

GetTypeFromProgID

```

22
23 [C#] public static Type GetTypeFromProgID(string progID, string server);
24 [C++] public: static Type* GetTypeFromProgID(String* progID, String* server);
25 [VB] Public Shared Function GetTypeFromProgID(ByVal progID As String,

```

ByVal server As String) As Type

[JScript] public static function GetTypeFromProgID(progID : String, server : String) : Type;

Description

Gets the **System.Type** associated with the specified program identifier (progID) from the specified server, returning null if an error is encountered while loading the **System.Type**.

Return Value: The **System.Type** associated with the specified program identifier (progID), if *progID* is a valid entry in the registry and a type is associated with it; otherwise, **null**.

This method is provided for COM support. Program IDs are not used in Microsoft .NET Framework because they have been superseded by the concept of namespace. The progID of the **System.Type** to get. The server from which to load the type.

GetTypeFromProgID

[C#] public static Type GetTypeFromProgID(string progID, string server, bool throwOnError);

[C++] public: static Type* GetTypeFromProgID(String* progID, String* server, bool throwOnError);

[VB] Public Shared Function GetTypeFromProgID(ByVal progID As String, ByVal server As String, ByVal throwOnError As Boolean) As Type

[JScript] public static function GetTypeFromProgID(progID : String, server : String, throwOnError : Boolean) : Type;

Description

Gets the **System.Type** associated with the specified program identifier (progID) from the specified server, specifying whether to throw an exception if an error occurs while loading the **System.Type**.

Return Value: The **System.Type** associated with the specified program identifier (progID), if *progID* is a valid entry in the registry and a type is associated with it; otherwise, **null**.

This method is provided for COM support. Program IDs are not used in Microsoft .NET Framework because they have been superseded by the concept of namespace. The progID of the **System.Type** to get. The server from which to load the type. **true** to throw a **System.TypeLoadException** if an error occurs while loading the **System.Type**.

GetTypeHandle

[C#] public static RuntimeTypeHandle GetTypeHandle(object o);

[C++] public: static RuntimeTypeHandle GetTypeHandle(Object* o);

[VB] Public Shared Function GetTypeHandle(ByVal o As Object) As

RuntimeTypeHandle

[JScript] public static function GetTypeHandle(o : Object) : RuntimeTypeHandle;

Description

Gets the handle for the **System.Type** of a specified object.

Return Value: The handle for the **System.Type** of the specified **System.Object**.

The handles are valid only in the application domain in which they were obtained. The **System.Object** for which to get the Type handle.

HasElementTypeImpl

[C#] protected abstract bool HasElementTypeImpl();

[C++] protected: virtual bool HasElementTypeImpl() = 0;

[VB] MustOverride Protected Function HasElementTypeImpl() As Boolean

[JScript] protected abstract function HasElementTypeImpl() : Boolean;

Description

When overridden in a derived class, implements the **System.Type.HasElementType** property and determines whether the current **System.Type** encompasses or refers to another type; that is, whether the current **System.Type** is an array, a pointer, or is passed by reference.

Return Value: **true** if the **System.Type** is an array, a pointer, or is passed by reference; otherwise, **false**.

For example, `Type.GetType("Int32[]").HasElementTypeImpl` returns **true**, but `Type.GetType("Int32").HasElementTypeImpl` returns **false**. `HasElementTypeImpl` also returns **true** for `"Int32*"` and `"Int32&"`.

InvokeMember

[C#] public object InvokeMember(string name, BindingFlags invokeAttr, Binder binder, object target, object[] args);

[C++] public: Object* InvokeMember(String* name, BindingFlags invokeAttr, Binder* binder, Object* target, Object* args __gc[]);

1 [VB] Public Function InvokeMember(ByVal name As String, ByVal invokeAttr
2 As BindingFlags, ByVal binder As Binder, ByVal target As Object, ByVal args()
3 As Object) As Object

4 [JScript] public function InvokeMember(name : String, invokeAttr : BindingFlags,
5 binder : Binder, target : Object, args : Object[]) : Object;

7 *Description*

8 Invokes the specified member, using the specified binding constraints and
9 matching the specified argument list.

10 *Return Value:* An **System.Object** representing the return value of the invoked
11 member.

12 The following **System.Reflection.BindingFlags** filter flags can be used to
13 define which members to include in the search: *Instance* to include instance
14 members in the search. The **System.String** containing the name of the constructor,
15 method, property, or field member to invoke. A bitmask comprised of one or more
16 **System.Reflection.BindingFlags** that specify how the search is conducted. The
17 access can be one of the **BindingFlags** such as **Public**, **NonPublic**, **Private**,
18 **InvokeMethod**, **GetField**, and so on. The type of lookup need not be specified. If
19 the type of lookup is omitted, **BindingFlags.DefaultLookup** will apply. A
20 **System.Reflection.Binder** object that defines a set of properties and enables
21 binding, which can involve selection of an overloaded method, coercion of
22 argument types, and invocation of a member through reflection. The
23 **System.Object** on which to invoke the specified member. An array containing the
24 arguments to pass to the member to invoke.

25 InvokeMember

1
2 [C#] public object InvokeMember(string name, BindingFlags invokeAttr, Binder
3 binder, object target, object[] args, CultureInfo culture);

4 [C++] public: Object* InvokeMember(String* name, BindingFlags invokeAttr,
5 Binder* binder, Object* target, Object* args __gc[], CultureInfo* culture);

6 [VB] Public Function InvokeMember(ByVal name As String, ByVal invokeAttr
7 As BindingFlags, ByVal binder As Binder, ByVal target As Object, ByVal args()
8 As Object, ByVal culture As CultureInfo) As Object

9 [JScript] public function InvokeMember(name : String, invokeAttr : BindingFlags,
10 binder : Binder, target : Object, args : Object[], culture : CultureInfo) : Object;

11 12 *Description*

13 Invokes the specified member, using the specified binding constraints and
14 matching the specified argument list and culture.

15 *Return Value:* An **System.Object** representing the return value of the invoked
16 member.

17 The following **System.Reflection.BindingFlags** filter flags can be used to
18 define which members to include in the search: *Instance* to include instance
19 members in the search. The **System.String** containing the name of the constructor,
20 method, property, or field member to invoke. A bitmask comprised of one or more
21 **System.Reflection.BindingFlags** that specify how the search is conducted. The
22 access can be one of the **BindingFlags** such as **Public**, **NonPublic**, **Private**,
23 **InvokeMethod**, **GetField**, and so on. The type of lookup need not be specified. If
24 the type of lookup is omitted, **BindingFlags.DefaultLookup** will apply. A
25 **System.Reflection.Binder** object that defines a set of properties and enables

binding, which can involve selection of an overloaded method, coercion of argument types, and invocation of a member through reflection. The **System.Object** on which to invoke the specified member. An array containing the arguments to pass to the member to invoke. The **System.Globalization.CultureInfo** object representing the globalization locale to use, which may be necessary for locale-specific conversions, such as converting a numeric String to a Double.

InvokeMember

[C#] public abstract object InvokeMember(string name, BindingFlags invokeAttr, Binder binder, object target, object[] args, ParameterModifier[] modifiers, CultureInfo culture, string[] namedParameters);

[C++] public: virtual Object* InvokeMember(String* name, BindingFlags invokeAttr, Binder* binder, Object* target, Object* args __gc[], ParameterModifier modifiers[], CultureInfo* culture, String* namedParameters __gc[]) = 0;

[VB] MustOverride Public Function InvokeMember(ByVal name As String, ByVal invokeAttr As BindingFlags, ByVal binder As Binder, ByVal target As Object, ByVal args() As Object, ByVal modifiers() As ParameterModifier, ByVal culture As CultureInfo, ByVal namedParameters() As String) As Object

[JScript] public abstract function InvokeMember(name : String, invokeAttr : BindingFlags, binder : Binder, target : Object, args : Object[], modifiers : ParameterModifier[], culture : CultureInfo, namedParameters : String[]) : Object;

Invokes a specific member of the current **System.Type**.

Description

When overridden in a derived class, invokes the specified member, using the specified binding constraints and matching the specified argument list, modifiers and culture.

Return Value: An **System.Object** representing the return value of the invoked member.

InvokeMember calls a constructor member or a method member, gets or sets a property member, gets or sets a data field member, or gets or sets an element of an array member. The **System.String** containing the name of the constructor, method, property, or field member to invoke. A bitmask comprised of one or more **System.Reflection.BindingFlags** that specify how the search is conducted. The access can be one of the **BindingFlags** such as **Public**, **NonPublic**, **Private**, **InvokeMethod**, **GetField**, and so on. The type of lookup need not be specified. If the type of lookup is omitted, **BindingFlags.DefaultLookup** will apply. A **System.Reflection.Binder** object that defines a set of properties and enables binding, which can involve selection of an overloaded method, coercion of argument types, and invocation of a member through reflection. The **System.Object** on which to invoke the specified member. An array containing the arguments to pass to the member to invoke. An array of **System.Reflection.ParameterModifier** objects representing the attributes associated with the corresponding element in the *args* array. A parameter's associated attributes are stored in the member's signature. The default binder does exact matching on the **System.Reflection.ParameterAttributesOut** and **None** attributes. The **System.Globalization.CultureInfo** object representing the

globalization locale to use, which may be necessary for locale-specific conversions, such as converting a numeric String to a Double. An array containing the names of the parameters to which the values in the *args* array are passed.

IsArrayImpl

[C#] protected abstract bool IsArrayImpl();

[C++] protected: virtual bool IsArrayImpl() = 0;

[VB] MustOverride Protected Function IsArrayImpl() As Boolean

[JScript] protected abstract function IsArrayImpl() : Boolean;

Description

When overridden in a derived class, implements the **System.Type.IsArray** property and determines whether the **System.Type** is an array.

Return Value: **true** if the **System.Type** is an array; otherwise, **false**.

An instance of the **System.Array** class must return **false** because it is an object, not an array.

IsAssignableFrom

[C#] public virtual bool IsAssignableFrom(Type c);

[C++] public: virtual bool IsAssignableFrom(Type* c);

[VB] Overridable Public Function IsAssignableFrom(ByVal c As Type) As Boolean

[JScript] public function IsAssignableFrom(c : Type) : Boolean;

Description

Determines whether an instance of the specified type can be assigned to the current **System.Type** instance.

Return Value: **true** if an instance of *c* can be assigned to the current **System.Type** instance; otherwise, **false** .

This method can be overridden by a derived class. The **System.Type** to examine whether its objects can be assigned to the current **System.Type** instance.

IsByRefImpl

[C#] protected abstract bool IsByRefImpl();

[C++] protected: virtual bool IsByRefImpl() = 0;

[VB] MustOverride Protected Function IsByRefImpl() As Boolean

[JScript] protected abstract function IsByRefImpl() : Boolean;

Description

When overridden in a derived class, implements the **System.Type.IsByRef** property and determines whether the **System.Type** is passed by reference.

Return Value: **true** if the **System.Type** is passed by reference; otherwise, **false** .

IsCOMObjectImpl

[C#] protected abstract bool IsCOMObjectImpl();

[C++] protected: virtual bool IsCOMObjectImpl() = 0;

[VB] MustOverride Protected Function IsCOMObjectImpl() As Boolean

[JScript] protected abstract function IsCOMObjectImpl() : Boolean;

Description

When overridden in a derived class, implements the **System.Type.IsCOMObject** property and determines whether the **System.Type** is a COM object.

Return Value: **true** if the **System.Type** is a COM object; otherwise, **false** .

This method returns **false** for COM interfaces because they are not objects. COM interfaces can be implemented by Microsoft .NET Framework objects.

IsContextfulImpl

[C#] protected virtual bool IsContextfulImpl();

[C++] protected: virtual bool IsContextfulImpl();

[VB] Overridable Protected Function IsContextfulImpl() As Boolean

[JScript] protected function IsContextfulImpl() : Boolean;

Description

Implements the **System.Type.IsContextful** property and determines whether the **System.Type** can be hosted in a context.

Return Value: **true** if the **System.Type** can be hosted in a context; otherwise, **false**

.

This method can be overridden by a derived class.

IsInstanceOfType

[C#] public virtual bool IsInstanceOfType(object o);

[C++] public: virtual bool IsInstanceOfType(Object* o);

[VB] Overridable Public Function IsInstanceOfType(ByVal o As Object) As

Boolean

1 [JScript] public function IsInstanceOfType(o : Object) : Boolean;

2
3 *Description*

4 Determines whether the specified object is an instance of the **System.Type**

5 .

6 *Return Value:* **true** if *o* is an instance of the **System.Type** ; otherwise, **false** .

7 This method can be overridden by a derived class. The **System.Object**
8 whose type to compare with **System.Type**.

9 IsMarshalByRefImpl

10
11 [C#] protected virtual bool IsMarshalByRefImpl();

12 [C++] protected: virtual bool IsMarshalByRefImpl();

13 [VB] Overridable Protected Function IsMarshalByRefImpl() As Boolean

14 [JScript] protected function IsMarshalByRefImpl() : Boolean;

15
16 *Description*

17 Implements the **System.Type.IsMarshalByRef** property and determines
18 whether the **System.Type** is marshalled by reference.

19 *Return Value:* **true** if the **System.Type** is marshalled by reference; otherwise,
20 **false** .

21 This method can be overridden by a derived class.

22 IsPointerImpl

23
24 [C#] protected abstract bool IsPointerImpl();

25 [C++] protected: virtual bool IsPointerImpl() = 0;

1 [VB] MustOverride Protected Function IsPointerImpl() As Boolean

2 [JScript] protected abstract function IsPointerImpl() : Boolean;

3
4 *Description*

5 When overridden in a derived class, implements the
6 **System.Type.IsPointer** property and determines whether the **System.Type** is a
7 pointer.

8 *Return Value:* **true** if the **System.Type** is a pointer; otherwise, **false** .

9 **IsPrimitiveImpl**

10
11 [C#] protected abstract bool IsPrimitiveImpl();

12 [C++] protected: virtual bool IsPrimitiveImpl() = 0;

13 [VB] MustOverride Protected Function IsPrimitiveImpl() As Boolean

14 [JScript] protected abstract function IsPrimitiveImpl() : Boolean;

15
16 *Description*

17 When overridden in a derived class, implements the
18 **System.Type.IsPrimitive** property and determines whether the **System.Type** is
19 one of the primitive types.

20 *Return Value:* **true** if the **System.Type** is one of the primitive types; otherwise,
21 **false** .

22 The primitive types are **System.Boolean** , **System.Byte** , **System.SByte** ,
23 **System.Int16** , **System.UInt16** , **System.Int32** , **System.UInt32** , **System.Int64** ,
24 **System.UInt64** , **System.Char** , **System.Double** , and **System.Single** .

25 **IsSubclassOf**

1
2 [C#] public virtual bool IsSubclassOf(Type c);

3 [C++] public: virtual bool IsSubclassOf(Type* c);

4 [VB] Overridable Public Function IsSubclassOf(ByVal c As Type) As Boolean

5 [JScript] public function IsSubclassOf(c : Type) : Boolean;

6
7 *Description*

8 Determines whether the current **System.Type** is a derived class of the
9 specified class.

10 *Return Value:* **true** if the current **System.Type** is a direct or indirect derived class
11 of *c* ; otherwise, **false** .

12 This method can be overridden by a derived class. The **System.Type** that
13 might be a base class of the current **System.Type**.

14 **IsValueTypeImpl**

15
16 [C#] protected virtual bool IsValueTypeImpl();

17 [C++] protected: virtual bool IsValueTypeImpl();

18 [VB] Overridable Protected Function IsValueTypeImpl() As Boolean

19 [JScript] protected function IsValueTypeImpl() : Boolean;

20
21 *Description*

22 Implements the **System.Type.IsValueType** property and determines
23 whether the **System.Type** is a value type; that is, not a class or an interface.

24 *Return Value:* **true** if the **System.Type** is a value type; otherwise, **false** .
25

Value types describe values that are represented as sequences of bits; value types are not classes or interfaces. These are referred to as "structs" in some programming languages. Enums are value types.

ToString

[C#] public override string ToString();

[C++] public: String* ToString();

[VB] Overrides Public Function ToString() As String

[JScript] public override function ToString() : String;

Description

Returns a **String** representing the name of the current **Type**.

Return Value: A **System.String** representing the name of the current **System.Type**.

This method returns the fully qualified common language runtime namespace and name for all primitive types. For example, the C# instruction, (long)0.Type().ToString() returns "System.Int64" instead of merely "Int64".

TypeCode enumeration (System)

ToString

Description

Specifies the type of an object.

1 Call the **System.IConvertible.GetTypeCode** method on classes that
2 implement the **System.IConvertible** interface to obtain the type code for an
3 instance of that class.

4 ToString

6 [C#] public const TypeCode Boolean;

7 [C++] public: const TypeCode Boolean;

8 [VB] Public Const Boolean As TypeCode

9 [JScript] public var Boolean : TypeCode;

11 *Description*

12 A simple type representing Boolean values of **true** or **false** .

13 ToString

15 [C#] public const TypeCode Byte;

16 [C++] public: const TypeCode Byte;

17 [VB] Public Const Byte As TypeCode

18 [JScript] public var Byte : TypeCode;

20 *Description*

21 An integral type representing unsigned 8-bit integers with values between 0
22 and 255.

23 ToString

25 [C#] public const TypeCode Char;

[C++] public: const TypeCode Char;
 [VB] Public Const Char As TypeCode
 [JScript] public var Char : TypeCode;

Description

An integral type representing unsigned 16-bit integers with values between 0 and 65535. The set of possible values for the **System.TypeCode.Char** type corresponds to the Unicode character set.

ToString

[C#] public const TypeCode DateTime;
 [C++] public: const TypeCode DateTime;
 [VB] Public Const DateTime As TypeCode
 [JScript] public var DateTime : TypeCode;

Description

A type representing a date and time value.

ToString

[C#] public const TypeCode DBNull;
 [C++] public: const TypeCode DBNull;
 [VB] Public Const DBNull As TypeCode
 [JScript] public var DBNull : TypeCode;

Description

1 A database null (column) value.

2 ToString

3
4 [C#] public const TypeCode Decimal;

5 [C++] public: const TypeCode Decimal;

6 [VB] Public Const Decimal As TypeCode

7 [JScript] public var Decimal : TypeCode;

8
9 *Description*

10 A simple type representing values ranging from 1.0×10 to approximately
11 7.9×10 with 28-29 significant digits.

12 ToString

13
14 [C#] public const TypeCode Double;

15 [C++] public: const TypeCode Double;

16 [VB] Public Const Double As TypeCode

17 [JScript] public var Double : TypeCode;

18
19 *Description*

20 A floating point type representing values ranging from approximately $5.0 \times$
21 10 to 1.7×10 with a precision of 15-16 digits.

22 ToString

23
24 [C#] public const TypeCode Empty;

25 [C++] public: const TypeCode Empty;

1 [VB] Public Const Empty As TypeCode

2 [JScript] public var Empty : TypeCode;

3
4 *Description*

5 A null reference.

6 ToString

7
8 [C#] public const TypeCode Int16;

9 [C++] public: const TypeCode Int16;

10 [VB] Public Const Int16 As TypeCode

11 [JScript] public var Int16 : TypeCode;

12
13 *Description*

14 An integral type representing signed 16-bit integers with values between -
15 32768 and 32767.

16 ToString

17
18 [C#] public const TypeCode Int32;

19 [C++] public: const TypeCode Int32;

20 [VB] Public Const Int32 As TypeCode

21 [JScript] public var Int32 : TypeCode;

22
23 *Description*

24 An integral type representing signed 32-bit integers with values between -
25 2147483648 and 2147483647.

ToString

[C#] public const TypeCode Int64;
[C++] public: const TypeCode Int64;
[VB] Public Const Int64 As TypeCode
[JScript] public var Int64 : TypeCode;

Description

An integral type representing signed 64-bit integers with values between -9223372036854775808 and 9223372036854775807.

ToString

[C#] public const TypeCode Object;
[C++] public: const TypeCode Object;
[VB] Public Const Object As TypeCode
[JScript] public var Object : TypeCode;

Description

A general type representing any reference or value type not explicitly represented by another **TypeCode**.

ToString

[C#] public const TypeCode SByte;
[C++] public: const TypeCode SByte;
[VB] Public Const SByte As TypeCode

1 [JScript] public var SByte : TypeCode;

3 *Description*

4 An integral type representing signed 8-bit integers with values between -
5 128 and 127.

6 ToString

8 [C#] public const TypeCode Single;

9 [C++] public: const TypeCode Single;

10 [VB] Public Const Single As TypeCode

11 [JScript] public var Single : TypeCode;

13 *Description*

14 A floating point type representing values ranging from approximately 1.5 x
15 10 to 3.4 x 10 with a precision of 7 digits.

16 ToString

18 [C#] public const TypeCode String;

19 [C++] public: const TypeCode String;

20 [VB] Public Const String As TypeCode

21 [JScript] public var String : TypeCode;

23 *Description*

24 A sealed class type representing Unicode character strings.

25 ToString

1
2 [C#] public const TypeCode UInt16;
3 [C++] public: const TypeCode UInt16;
4 [VB] Public Const UInt16 As TypeCode
5 [JScript] public var UInt16 : TypeCode;

6
7 *Description*

8 An integral type representing unsigned 16-bit integers with values between
9 0 and 65535.

10 ToString

11
12 [C#] public const TypeCode UInt32;
13 [C++] public: const TypeCode UInt32;
14 [VB] Public Const UInt32 As TypeCode
15 [JScript] public var UInt32 : TypeCode;

16
17 *Description*

18 An integral type representing unsigned 32-bit integers with values between
19 0 and 4294967295.

20 ToString

21
22 [C#] public const TypeCode UInt64;
23 [C++] public: const TypeCode UInt64;
24 [VB] Public Const UInt64 As TypeCode
25 [JScript] public var UInt64 : TypeCode;

1
2 *Description*

3 An integral type representing unsigned 64-bit integers with values between
4 0 and 18446744073709551615.

5 TypedReference structure (System)

6 ToString
7
8

9 *Description*

10 Describes objects that contain both a managed pointer to a location and a
11 runtime representation of the type that may be stored at that location.

12 A typed reference is a type/value combination used for varargs and other
13 support.

14 Equals
15

16 [C#] public override bool Equals(object o);

17 [C++] public: bool Equals(Object* o);

18 [VB] Overrides Public Function Equals(ByVal o As Object) As Boolean

19 [JScript] public override function Equals(o : Object) : Boolean;
20

21 *Description*

22 Checks if this object is equal to the specified object.

23 *Return Value:* **true** if this object is equal to the specified object; otherwise, **false** .

24 The object with which to compare the current object.

25 GetHashCode

1
2 [C#] public override int GetHashCode();

3 [C++] public: int GetHashCode();

4 [VB] Overrides Public Function GetHashCode() As Integer

5 [JScript] public override function GetHashCode() : int;

6
7 *Description*

8 Returns the hash code of this object.

9 *Return Value:* The hash code of this object.

10 **GetTargetType**

11
12 [C#] public static Type GetTargetType(TypedReference value);

13 [C++] public: static Type* GetTargetType(TypedReference value);

14 [VB] Public Shared Function GetTargetType(ByVal value As TypedReference)

15 As Type

16 [JScript] public static function GetTargetType(value : TypedReference) : Type;

17
18 *Description*

19 Returns the type of the target of the specified **TypedReference** .

20 *Return Value:* The type of the target of the specified **TypedReference** . The value
21 whose target's type is to be returned.

22 **MakeTypedReference**

23
24 [C#] public static TypedReference MakeTypedReference(object target, FieldInfo[]

25 flds);

```

1 [C++] public: static TypedReference MakeTypedReference(Object* target,
2   FieldInfo* flds[]);
3 [VB] Public Shared Function MakeTypedReference(ByVal target As Object,
4   ByVal flds() As FieldInfo) As TypedReference
5 [JScript] public static function MakeTypedReference(target : Object, flds :
6   FieldInfo[]) : TypedReference;

```

Description

Makes a **TypedReference** for the specified target object using the specifying fields.

Return Value: A **TypedReference** for the specified target. The target object that defines the type of the **TypedReference** . The fields to be encapsulated.

SetTypedReference

```

15 [C#] public static void SetTypedReference(TypedReference target, object value);
16 [C++] public: static void SetTypedReference(TypedReference target, Object*
17   value);
18 [VB] Public Shared Sub SetTypedReference(ByVal target As TypedReference,
19   ByVal value As Object)
20 [JScript] public static function SetTypedReference(target : TypedReference, value
21   : Object);

```

Description

Converts the specified value to a **TypedReference** .

Return Value: This method assigns *value* to *target* . A change type of *value*

converts it to the type of the **TypedReference** . The

System.Convert.ChangeType(System.Object,System.TypeCode) method does the conversion. The target of the conversion. The value to be converted.

TargetTypeToken

[C#] public static RuntimeTypeHandle TargetTypeToken(TypedReference value);

[C++] public: static RuntimeTypeHandle TargetTypeToken(TypedReference value);

[VB] Public Shared Function TargetTypeToken(ByVal value As TypedReference) As RuntimeTypeHandle

[JScript] public static function TargetTypeToken(value : TypedReference) : RuntimeTypeHandle;

Description

Returns the internal metadata type handle for the specified

TypedReference .

Return Value: The internal metadata type handle for the specified

TypedReference . The **TypedReference** for which the type handle is requested.

ToObject

[C#] public static object ToObject(TypedReference value);

[C++] public: static Object* ToObject(TypedReference value);

[VB] Public Shared Function ToObject(ByVal value As TypedReference) As Object

[JScript] public static function ToObject(value : TypedReference) : Object;

1
2 *Description*

3 Converts the specified **TypedReference** to an **Object** .

4 *Return Value:* An Object converted from a TypedReference.

5 This might be a boxing operation. The **TypedReference** to be converted.

6 TypeInitializationException class (System)

7 ToString

8
9
10 *Description*

11 The exception that is thrown as a wrapper around the exception thrown by
12 the class initializer. This class cannot be inherited.

13 When a class initializer fails to initialize a type, a

14 **System.TypeInitializationException** is created and passed a reference to the
15 exception thrown by the type's class initializer. The

16 **System.Exception.InnerException** property of

17 **System.TypeInitializationException** holds the underlying exception.

18 TypeInitializationException

19 *Example Syntax:*

20 ToString

21
22 [C#] public TypeInitializationException(string fullTypeName, Exception
23 innerException);

24 [C++] public: TypeInitializationException(String* fullTypeName, Exception*
25 innerException);

[VB] Public Sub New(ByVal fullTypeName As String, ByVal innerException As Exception)

[JScript] public function TypeInitializationException(fullTypeName : String, innerException : Exception);

Description

Initializes a new instance of the **System.TypeInitializationException** class with the default error message, the specified type name, and a reference to the inner exception that is the root cause of this exception.

When an **Exception** *X* is thrown as a direct result of a previous exception *Y*, the **System.Exception.InnerException** property of *X* should contain a reference to *Y*. The **InnerException** property returns the same value as was passed into the constructor, or **null** if the inner exception value was not supplied to the constructor. The fully qualified name of the type that fails to initialize. An instance of **System.Exception** that is the cause of the current **Exception**. If *innerException* is non-null, then the current **Exception** is raised in a catch block handling *innerException*.

HelpLink

HResult

InnerException

Message

Source

StackTrace

TargetSite

TypeName

ToString

Description

Gets the fully qualified name of the type that fails to initialize.

TypeLoadException class (System)

ToString

Description

The exception that is thrown when type-loading failures occur.

System.TypeLoadException is thrown when the Common Language Runtime cannot find the assembly, the type within the assembly, or cannot load the type.

TypeLoadException

Example Syntax:

ToString

[C#] public TypeLoadException();

[C++] public: TypeLoadException();

[VB] Public Sub New()

[JScript] public function TypeLoadException(); Initializes a new instance of the

System.TypeLoadException class.

Description

1 Initializes a new instance of the **System.TypeLoadException** class with
2 default properties.

3 The following table shows the initial property values for an instance of
4 **System.TypeLoadException** .

5 TypeLoadException

6 *Example Syntax:*

7 ToString

9 [C#] public TypeLoadException(string message);

10 [C++] public: TypeLoadException(String* message);

11 [VB] Public Sub New(ByVal message As String)

12 [JScript] public function TypeLoadException(message : String);

14 *Description*

15 Initializes a new instance of the **System.TypeLoadException** class with a
16 specified error message.

17 The following table shows the initial property values for an instance of
18 **System.TypeLoadException** . The error message that explains the reason for the
19 exception.

20 TypeLoadException

21 *Example Syntax:*

22 ToString

24 [C#] protected TypeLoadException(SerializationInfo info, StreamingContext
25 context);

1 [C++] protected: TypeLoadException(SerializationInfo* info, StreamingContext
2 context);

3 [VB] Protected Sub New(ByVal info As SerializationInfo, ByVal context As
4 StreamingContext)

5 [JScript] protected function TypeLoadException(info : SerializationInfo, context :
6 StreamingContext);

7 8 *Description*

9 Initializes a new instance of the **System.TypeLoadException** class with
10 serialized data.

11 This constructor is called during deserialization to reconstitute the
12 exception object transmitted over a stream. For more information, see . The object
13 that holds the serialized object data. The contextual information about the source
14 or destination.

15 TypeLoadException

16 *Example Syntax:*

17 ToString

18
19 [C#] public TypeLoadException(string message, Exception inner);

20 [C++] public: TypeLoadException(String* message, Exception* inner);

21 [VB] Public Sub New(ByVal message As String, ByVal inner As Exception)

22 [JScript] public function TypeLoadException(message : String, inner : Exception);

23 24 *Description*

1 Initializes a new instance of the **System.TypeLoadException** class with a
2 specified error message and a reference to the inner exception that is the root cause
3 of this exception.

4 When an **Exception** *X* is thrown as a direct result of a previous exception *Y* ,
5 the **System.Exception.InnerException** property of *X* should contain a reference
6 to *Y* . The **InnerException** property returns the same value as was passed into the
7 constructor, or **null** if the inner exception value was not supplied to the
8 constructor. The error message that explains the reason for the exception. An
9 instance of **System.Exception** that is the cause of the current **Exception**. If *inner*
10 is non-null, then the current **Exception** is raised in a catch block handling *inner* .

11 HelpLink

12 HResult

13 InnerException

14 Message

15 ToString

16
17
18 *Description*

19 Gets the error message for this exception.

20 This property overrides **System.Exception.Message** . The error message
21 should be localized.

22 Source

23 StackTrace

24 TargetSite

25 TypeName

ToString

Description

Gets the fully qualified name of the type that causes the exception.

When overriding **System.TypeLoadException.TypeName** in a derived class, be sure to call the base class's **System.TypeLoadException.TypeName** property.

GetObjectData

[C#] public override void GetObjectData(SerializationInfo info, StreamingContext context);

[C++] public: void GetObjectData(SerializationInfo* info, StreamingContext context);

[VB] Overrides Public Sub GetObjectData(ByVal info As SerializationInfo, ByVal context As StreamingContext)

[JScript] public override function GetObjectData(info : SerializationInfo, context : StreamingContext);

Description

Sets the **System.Runtime.Serialization.SerializationInfo** object with the class name, method name, resource ID, and additional exception information.

System.TypeLoadException.GetObjectData(System.Runtime.Serialization.SerializationInfo, System.Runtime.Serialization.StreamingContext) sets a **System.Runtime.Serialization.SerializationInfo** with all the exception object

data targeted for serialization. During deserialization, the exception object is reconstituted from the **System.Runtime.Serialization.SerializationInfo** transmitted over the stream. The object that holds the serialized object data. The contextual information about the source or destination.

TypeUnloadedException class (System)

ToString

Description

The exception that is thrown when there is an attempt to access an unloaded class.

System.TypeUnloadedException uses the HRESULT **COR_E_TYPEUNLOADED**, which has the value 0x80131013.

TypeUnloadedException

Example Syntax:

ToString

[C#] public TypeUnloadedException();

[C++] public: TypeUnloadedException();

[VB] Public Sub New()

[JScript] public function TypeUnloadedException(); Initializes new instance of the **System.TypeUnloadedException** class.

Description

1 Initializes a new instance of the **System.TypeUnloadedException** class
2 with default properties.

3 The following table shows the initial property values for an instance of
4 **System.TypeUnloadedException** .

5 TypeUnloadedException

6 *Example Syntax:*

7 ToString

9 [C#] public TypeUnloadedException(string message);

10 [C++] public: TypeUnloadedException(String* message);

11 [VB] Public Sub New(ByVal message As String)

12 [JScript] public function TypeUnloadedException(message : String);

14 *Description*

15 Initializes a new instance of the **System.TypeUnloadedException** class
16 with a specified error message.

17 The following table shows the initial property values for an instance of
18 **System.TypeUnloadedException** . The error message that explains the reason for
19 the exception.

20 TypeUnloadedException

21 *Example Syntax:*

22 ToString

24 [C#] protected TypeUnloadedException(SerializationInfo info, StreamingContext
25 context);

1 [C++] protected: TypeUnloadedException(SerializationInfo* info,
 2 StreamingContext context);
 3 [VB] Protected Sub New(ByVal info As SerializationInfo, ByVal context As
 4 StreamingContext)
 5 [JScript] protected function TypeUnloadedException(info : SerializationInfo,
 6 context : StreamingContext);

8 *Description*

9 Initializes a new instance of the **System.TypeUnloadedException** class
 10 with serialized data.

11 This constructor is called during deserialization to reconstitute the
 12 exception object transmitted over a stream. For more information, see . The object
 13 that holds the serialized object data. The contextual information about the source
 14 or destination.

15 TypeUnloadedException

16 *Example Syntax:*

17 ToString

18
 19 [C#] public TypeUnloadedException(string message, Exception innerException);
 20 [C++] public: TypeUnloadedException(String* message, Exception*
 21 innerException);
 22 [VB] Public Sub New(ByVal message As String, ByVal innerException As
 23 Exception)
 24 [JScript] public function TypeUnloadedException(message : String,
 25 innerException : Exception);

Description

Initializes a new instance of the **System.TypeUnloadedException** class with a specified error message and a reference to the inner exception that is the root cause of this exception.

When an **Exception** *X* is thrown as a direct result of a previous exception *Y*, the **System.Exception.InnerException** property of *X* should contain a reference to *Y*. The **InnerException** property returns the same value as was passed into the constructor, or **null** if the inner exception value was not supplied to the constructor. The error message that explains the reason for the exception. An instance of **System.Exception** that is the cause of the current **Exception**. If *innerException* is non-null, then the current **Exception** is raised in a catch block handling *innerException*.

HelpLink

HResult

InnerException

Message

Source

StackTrace

TargetSite

UInt16 structure (System)

ToString

Description

Represents a 16-bit unsigned integer.

The **UInt16** value type represents unsigned integers with values ranging from 0 to 65535.

ToString

[C#] public const ushort MaxValue;

[C++] public: const unsigned short MaxValue;

[VB] Public Const MaxValue As UInt16

[JScript] public var MaxValue : UInt16;

Description

A constant representing the largest possible value of **UInt16**.

The value of this constant is 65535; that is, hexadecimal 0xFFFF.

ToString

[C#] public const ushort MinValue;

[C++] public: const unsigned short MinValue;

[VB] Public Const MinValue As UInt16

[JScript] public var MinValue : UInt16;

Description

A constant representing the smallest possible value of **UInt16**.

The value of this constant is 0.

CompareTo

1
2 [C#] public int CompareTo(object value);

3 [C++] public: __sealed int CompareTo(Object* value);

4 [VB] NotOverridable Public Function CompareTo(ByVal value As Object) As

5 Integer

6 [JScript] public function CompareTo(value : Object) : int;

7
8 *Description*

9 Compares this instance to a specified object and returns an indication of
10 their relative values.

11 *Return Value:* A signed number indicating the relative values of this instance and
12 *value* .

13 Any instance of **UInt16** , regardless of its value, is considered greater than
14 **null** . An object to compare, or **null**.

15 *Equals*

16
17 [C#] public override bool Equals(object obj);

18 [C++] public: bool Equals(Object* obj);

19 [VB] Overrides Public Function Equals(ByVal obj As Object) As Boolean

20 [JScript] public override function Equals(obj : Object) : Boolean;

21
22 *Description*

23 Returns a value indicating whether this instance is equal to a specified
24 object.

Return Value: **true** if *obj* is an instance of **UInt16** and equals the value of this instance; otherwise, **false** . An object to compare with this instance.

GetHashCode

[C#] public override int GetHashCode();

[C++] public: int GetHashCode();

[VB] Overrides Public Function GetHashCode() As Integer

[JScript] public override function GetHashCode() : int;

Description

Returns the hash code for this instance.

Return Value: A 32-bit signed integer hash code.

GetTypeCode

[C#] public TypeCode GetTypeCode();

[C++] public: __sealed TypeCode GetTypeCode();

[VB] NotOverridable Public Function GetTypeCode() As TypeCode

[JScript] public function GetTypeCode() : TypeCode;

Description

Returns the **TypeCode** for value type **UInt16** .

Return Value: The enumerated constant, **System.TypeCode.UInt16** .

Parse

[C#] public static ushort Parse(string s);

1 [C++] public: static unsigned short Parse(String* s);

2 [VB] Public Shared Function Parse(ByVal s As String) As UInt16

3 [JScript] public static function Parse(s : String) : UInt16; Converts the **String**
4 representation of a number to its 16-bit unsigned integer equivalent.

5
6 *Description*

7 Converts the **String** representation of a number to its 16-bit unsigned
8 integer equivalent.

9 *Return Value:* An 16-bit unsigned integer equivalent to the number contained in *s* .

10 *s* contains a number of the form: [ws][sign]digits[ws] Items in square
11 brackets ('[' and ']') are optional, and other items are as follows. A **System.String**
12 containing a number to convert.

13 **Parse**

14
15 [C#] public static ushort Parse(string s, IFormatProvider provider);

16 [C++] public: static unsigned short Parse(String* s, IFormatProvider* provider);

17 [VB] Public Shared Function Parse(ByVal s As String, ByVal provider As
18 IFormatProvider) As UInt16

19 [JScript] public static function Parse(s : String, provider : IFormatProvider) :
20 UInt16;

21
22 *Description*

23 Converts the **String** representation of a number in a specified culture-
24 specific format to its 16-bit unsigned integer equivalent.

25 *Return Value:* An 16-bit unsigned integer equivalent to the number specified in *s* .

1 *s* contains a number of the form: [ws][sign]digits[ws] Items in square
2 brackets ('[' and ']') are optional, and other items are as follows. A **System.String**
3 containing a number to convert. An **System.IFormatProvider** interface
4 implementation which supplies culture-specific formatting information about *s*.

5 Parse

6
7 [C#] public static ushort Parse(string s, NumberStyles style);

8 [C++] public: static unsigned short Parse(String* s, NumberStyles style);

9 [VB] Public Shared Function Parse(ByVal s As String, ByVal style As
10 NumberStyles) As UInt16

11 [JScript] public static function Parse(s : String, style : NumberStyles) : UInt16;

12 Description

13
14 Converts the **String** representation of a number in a specified style to its
15 16-bit unsigned integer equivalent.

16 *Return Value:* An 16-bit unsigned integer equivalent to the number specified in *s* .

17 *s* contains a number of the form: [ws][sign]digits[ws] Items in square
18 brackets ('[' and ']') are optional, and other items are as follows. A **System.String**
19 containing a number to convert. The combination of one or more
20 **System.Globalization.NumberStyles** constants that indicate the permitted format
21 of *s*.

22 Parse

23
24 [C#] public static ushort Parse(string s, NumberStyles style, IFormatProvider
25 provider);


```

1 [C++] public: static unsigned short Parse(String* s, NumberStyles style,
2 IFormatProvider* provider);
3 [VB] Public Shared Function Parse(ByVal s As String, ByVal style As
4 NumberStyles, ByVal provider As IFormatProvider) As UInt16
5 [JScript] public static function Parse(s : String, style : NumberStyles, provider :
6 IFormatProvider) : UInt16;

```

Description

Converts the **String** representation of a number in a specified style and culture-specific format to its 16-bit unsigned integer equivalent.

Return Value: An 16-bit unsigned integer equivalent to the number specified in *s*.

s contains a number of the form: [ws][sign]digits[ws] Items in square brackets ('[' and ']') are optional, and other items are as follows. A **System.String** containing a number to convert. The combination of one or more **System.Globalization.NumberStyles** constants that indicate the permitted format of *s*. An **System.IFormatProvider** interface implementation which supplies culture-specific formatting information about *s*.

Convertible.ToBoolean

```

18
19
20 [C#] bool Convertible.ToBoolean(IFormatProvider provider);
21 [C++] bool Convertible::ToBoolean(IFormatProvider* provider);
22 [VB] Function ToBoolean(ByVal provider As IFormatProvider) As Boolean
23 Implements Convertible.ToBoolean
24 [JScript] function Convertible.ToBoolean(provider : IFormatProvider) : Boolean;
25
26 Convertible.ToByte

```

```

1
2 [C#] byte IConvertible.ToByte(IFormatProvider provider);
3 [C++] unsigned char IConvertible::ToByte(IFormatProvider* provider);
4 [VB] Function ToByte(ByVal provider As IFormatProvider) As Byte Implements
5 IConvertible.ToByte
6 [JScript] function IConvertible.ToByte(provider : IFormatProvider) : Byte;
7     IConvertible.ToChar
8
9 [C#] char IConvertible.ToChar(IFormatProvider provider);
10 [C++] __wchar_t IConvertible::ToChar(IFormatProvider* provider);
11 [VB] Function ToChar(ByVal provider As IFormatProvider) As Char Implements
12 IConvertible.ToChar
13 [JScript] function IConvertible.ToChar(provider : IFormatProvider) : Char;
14     IConvertible.ToDateTime
15
16 [C#] DateTime IConvertible.ToDateTime(IFormatProvider provider);
17 [C++] DateTime IConvertible::ToDateTime(IFormatProvider* provider);
18 [VB] Function ToDateTime(ByVal provider As IFormatProvider) As DateTime
19 Implements IConvertible.ToDateTime
20 [JScript] function IConvertible.ToDateTime(provider : IFormatProvider) :
21 DateTime;
22     IConvertible.ToDecimal
23
24 [C#] decimal IConvertible.ToDecimal(IFormatProvider provider);
25 [C++] Decimal IConvertible::ToDecimal(IFormatProvider* provider);
    
```

```

1 [VB] Function ToDecimal(ByVal provider As IFormatProvider) As Decimal
2 Implements IConvertible.ToDecimal
3 [JScript] function IConvertible.ToDecimal(provider : IFormatProvider) : Decimal;
4     IConvertible.ToDouble
5
6 [C#] double IConvertible.ToDouble(IFormatProvider provider);
7 [C++] double IConvertible::ToDouble(IFormatProvider* provider);
8 [VB] Function ToDouble(ByVal provider As IFormatProvider) As Double
9 Implements IConvertible.ToDouble
10 [JScript] function IConvertible.ToDouble(provider : IFormatProvider) : double;
11     IConvertible.ToInt16
12
13 [C#] short IConvertible.ToInt16(IFormatProvider provider);
14 [C++] short IConvertible::ToInt16(IFormatProvider* provider);
15 [VB] Function ToInt16(ByVal provider As IFormatProvider) As Short
16 Implements IConvertible.ToInt16
17 [JScript] function IConvertible.ToInt16(provider : IFormatProvider) : Int16;
18     IConvertible.ToInt32
19
20 [C#] int IConvertible.ToInt32(IFormatProvider provider);
21 [C++] int IConvertible::ToInt32(IFormatProvider* provider);
22 [VB] Function ToInt32(ByVal provider As IFormatProvider) As Integer
23 Implements IConvertible.ToInt32
24 [JScript] function IConvertible.ToInt32(provider : IFormatProvider) : int;
25     IConvertible.ToInt64

```

```

1  [C#] long IConvertible.ToInt64(IFormatProvider provider);
2
3  [C++] __int64 IConvertible::ToInt64(IFormatProvider* provider);
4
5  [VB] Function ToInt64(ByVal provider As IFormatProvider) As Long Implements
6  IConvertible.ToInt64
7
8  [JScript] function IConvertible.ToInt64(provider : IFormatProvider) : long;
9
10     IConvertible.ToSByte
11
12
13
14
15
16 [C#] sbyte IConvertible.ToSByte(IFormatProvider provider);
17
18 [C++] char IConvertible::ToSByte(IFormatProvider* provider);
19
20 [VB] Function ToSByte(ByVal provider As IFormatProvider) As SByte
21 Implements IConvertible.ToSByte
22
23 [JScript] function IConvertible.ToSByte(provider : IFormatProvider) : SByte;
24
25     IConvertible.ToSingle
26
27
28
29
30
31 [C#] float IConvertible.ToSingle(IFormatProvider provider);
32
33 [C++] float IConvertible::ToSingle(IFormatProvider* provider);
34
35 [VB] Function ToSingle(ByVal provider As IFormatProvider) As Single
36 Implements IConvertible.ToSingle
37
38 [JScript] function IConvertible.ToSingle(provider : IFormatProvider) : float;
39
40     IConvertible.ToType
41
42
43
44
45
46 [C#] object IConvertible.ToType(Type type, IFormatProvider provider);
47
48 [C++] Object* IConvertible::ToType(Type* type, IFormatProvider* provider);
49
50 [VB] Function ToType(ByVal type As Type, ByVal provider As IFormatProvider)

```

```

1 As Object Implements IConvertible.ToType
2 [JScript] function IConvertible.ToType(type : Type, provider : IFormatProvider) :
3 Object;
4     IConvertible.ToUInt16
5
6 [C#] ushort IConvertible.ToUInt16(IFormatProvider provider);
7 [C++] unsigned short IConvertible::ToUInt16(IFormatProvider* provider);
8 [VB] Function ToUInt16(ByVal provider As IFormatProvider) As UInt16
9 Implements IConvertible.ToUInt16
10 [JScript] function IConvertible.ToUInt16(provider : IFormatProvider) : UInt16;
11     IConvertible.ToUInt32
12
13 [C#] uint IConvertible.ToUInt32(IFormatProvider provider);
14 [C++] unsigned int IConvertible::ToUInt32(IFormatProvider* provider);
15 [VB] Function ToUInt32(ByVal provider As IFormatProvider) As UInt32
16 Implements IConvertible.ToUInt32
17 [JScript] function IConvertible.ToUInt32(provider : IFormatProvider) : UInt32;
18     IConvertible.ToUInt64
19
20 [C#] ulong IConvertible.ToUInt64(IFormatProvider provider);
21 [C++] unsigned __int64 IConvertible::ToUInt64(IFormatProvider* provider);
22 [VB] Function ToUInt64(ByVal provider As IFormatProvider) As UInt64
23 Implements IConvertible.ToUInt64
24 [JScript] function IConvertible.ToUInt64(provider : IFormatProvider) : UInt64;
25     ToString

```

[C#] public override string ToString();

[C++] public: String* ToString();

[VB] Overrides Public Function ToString() As String

[JScript] public override function ToString() : String; Converts the numeric value of this instance to its equivalent **String** representation.

Description

Converts the numeric value of this instance to its equivalent **String** representation.

Return Value: The **System.String** representation of the value of this instance, consisting of a sequence of digits ranging from 0 to 9, without a sign or leading zeroes.

The return value is formatted with the general format specifier ("G") and the **System.Globalization.NumberFormatInfo** for the current culture.

ToString

[C#] public string ToString(IFormatProvider provider);

[C++] public: __sealed String* ToString(IFormatProvider* provider);

[VB] NotOverridable Public Function ToString(ByVal provider As IFormatProvider) As String

[JScript] public function ToString(provider : IFormatProvider) : String;

Description

Converts the numeric value of this instance to its equivalent **String** representation using the specified culture-specific format information.

Return Value: The **System.String** representation of the value of this instance as specified by *provider* .

This instance is formatted with the general format specifier ("G"). An **System.IFormatProvider** interface implementation which supplies culture-specific formatting information.

ToString

[C#] public string ToString(string format);

[C++] public: String* ToString(String* format);

[VB] Public Function ToString(ByVal format As String) As String

[JScript] public function ToString(format : String) : String;

Description

Converts the numeric value of this instance to its equivalent **String** representation using the specified format.

Return Value: The **System.String** representation of the value of this instance as specified by *format* .

If *format* is **null** or an empty string (""), the return value of this instance is formatted with the general format specifier ("G"). A format string.

ToString

[C#] public string ToString(string format, IFormatProvider provider);

[C++] public: __sealed String* ToString(String* format, IFormatProvider*

provider);

[VB] NotOverridable Public Function ToString(ByVal format As String, ByVal
provider As IFormatProvider) As String

[JScript] public function ToString(format : String, provider : IFormatProvider) :
String;

Description

Converts the numeric value of this instance to its equivalent **String**
representation using the specified format and culture-specific format information.

Return Value: The **System.String** representation of the value of this instance as
specified by *format* and *provider*.

If *format* is **null** or an empty string (""), the return value for this instance is
formatted with the general format specifier ("G"). A format specification. An
System.IFormatProvider interface implementation which supplies culture-
specific formatting information about this instance.

UInt32 structure (System)

ToString

Description

Represents a 32-bit unsigned integer.

The **UInt32** value type represents unsigned integers with values ranging
from 0 to 4,294,967,295.

ToString

1
2 [C#] public const uint MaxValue;
3 [C++] public: const unsigned int MaxValue;
4 [VB] Public Const MaxValue As UInt32
5 [JScript] public var MaxValue : UInt32;

6
7 *Description*

8 A constant representing the largest possible value of **UInt32** .
9 The value of this constant is 4294967295; that is, hexadecimal
10 0xFFFFFFFF.

11 ToString

12
13 [C#] public const uint MinValue;
14 [C++] public: const unsigned int MinValue;
15 [VB] Public Const MinValue As UInt32
16 [JScript] public var MinValue : UInt32;

17
18 *Description*

19 A constant representing the smallest possible value of **UInt32** .
20 The value of this constant is 0.

21 CompareTo

22
23 [C#] public int CompareTo(object value);
24 [C++] public: __sealed int CompareTo(Object* value);
25 [VB] NotOverridable Public Function CompareTo(ByVal value As Object) As

Integer

[JScript] public function CompareTo(value : Object) : int;

Description

Compares this instance to a specified object and returns an indication of their relative values.

Return Value: A signed number indicating the relative values of this instance and *value*.

Any instance of **UInt32**, regardless of its value, is considered greater than **null**. An object to compare, or **null**.

Equals

[C#] public override bool Equals(object obj);

[C++] public: bool Equals(Object* obj);

[VB] Overrides Public Function Equals(ByVal obj As Object) As Boolean

[JScript] public override function Equals(obj : Object) : Boolean;

Description

Returns a value indicating whether this instance is equal to a specified object.

Return Value: **true** if *obj* is an instance of **UInt32** and equals the value of this instance; otherwise, **false**. An object to compare with this instance.

GetHashCode

[C#] public override int GetHashCode();

1 [C++] public: int GetHashCode();
 2 [VB] Overrides Public Function GetHashCode() As Integer
 3 [JScript] public override function GetHashCode() : int;

4
 5 *Description*

6 Returns the hash code for this instance.
 7 *Return Value:* A 32-bit signed integer hash code.

8 GetTypeCode

9
 10 [C#] public TypeCode GetTypeCode();
 11 [C++] public: __sealed TypeCode GetTypeCode();
 12 [VB] NotOverridable Public Function GetTypeCode() As TypeCode
 13 [JScript] public function GetTypeCode() : TypeCode;

14
 15 *Description*

16 Returns the **TypeCode** for value type **UInt32** .
 17 *Return Value:* The enumerated constant, **System.TypeCode.UInt32** .

18 Parse

19
 20 [C#] public static uint Parse(string s);
 21 [C++] public: static unsigned int Parse(String* s);
 22 [VB] Public Shared Function Parse(ByVal s As String) As UInt32
 23 [JScript] public static function Parse(s : String) : UInt32; Converts the **String**
 24 representation of a number to its 32-bit unsigned integer equivalent.

25

Description

Converts the **String** representation of a number to its 32-bit unsigned integer equivalent.

Return Value: An 32-bit unsigned integer equivalent to the number contained in *s*.

s contains a number of the form: [ws][sign]digits[ws] Items in square brackets ('[' and ']') are optional, and other items are as follows. A **System.String** containing a number to convert.

Parse

[C#] public static uint Parse(string s, IFormatProvider provider);

[C++] public: static unsigned int Parse(String* s, IFormatProvider* provider);

[VB] Public Shared Function Parse(ByVal s As String, ByVal provider As IFormatProvider) As UInt32

[JScript] public static function Parse(s : String, provider : IFormatProvider) : UInt32;

Description

Converts the **String** representation of a number in a specified culture-specific format to its 32-bit unsigned integer equivalent.

Return Value: An 32-bit unsigned integer equivalent to the number specified in *s*.

s contains a number of the form: [ws][sign]digits[ws] Items in square brackets ('[' and ']') are optional, and other items are as follows. A **System.String** containing a number to convert. An **System.IFormatProvider** interface implementation which supplies culture-specific formatting information about *s*.

Parse

```
[C#] public static uint Parse(string s, NumberStyles style);  
[C++] public: static unsigned int Parse(String* s, NumberStyles style);  
[VB] Public Shared Function Parse(ByVal s As String, ByVal style As  
NumberStyles) As UInt32  
[JScript] public static function Parse(s : String, style : NumberStyles) : UInt32;
```

Description

Converts the **String** representation of a number in a specified style to its 32-bit unsigned integer equivalent.

Return Value: An 32-bit unsigned integer equivalent to the number specified in *s*.

s contains a number of the form: [ws][sign]digits[ws] Items in square brackets ('[' and ']') are optional, and other items are as follows. A **System.String** containing a number to convert. The combination of one or more **System.Globalization.NumberStyles** constants that indicate the permitted format of *s*.

Parse

```
[C#] public static uint Parse(string s, NumberStyles style, IFormatProvider  
provider);  
[C++] public: static unsigned int Parse(String* s, NumberStyles style,  
IFormatProvider* provider);  
[VB] Public Shared Function Parse(ByVal s As String, ByVal style As  
NumberStyles, ByVal provider As IFormatProvider) As UInt32
```

[JScript] public static function Parse(s : String, style : NumberStyles, provider :
IFormatProvider) : UInt32;

Description

Converts the **String** representation of a number in a specified style and culture-specific format to its 32-bit unsigned integer equivalent.

Return Value: An 32-bit unsigned integer equivalent to the number specified in *s*.

s contains a number of the form: [ws][sign]digits[ws] Items in square brackets ('[' and ']') are optional, and other items are as follows. A **System.String** containing a number to convert. The combination of one or more **System.Globalization.NumberStyles** constants that indicate the permitted format of *s*. An **System.IFormatProvider** interface implementation which supplies culture-specific formatting information about *s*.

IConvertible.ToBoolean

[C#] bool IConvertible.ToBoolean(IFormatProvider provider);

[C++] bool IConvertible::ToBoolean(IFormatProvider* provider);

[VB] Function ToBoolean(ByVal provider As IFormatProvider) As Boolean

Implements IConvertible.ToBoolean

[JScript] function IConvertible.ToBoolean(provider : IFormatProvider) : Boolean;

IConvertible.ToByte

[C#] byte IConvertible.ToByte(IFormatProvider provider);

[C++] unsigned char IConvertible::ToByte(IFormatProvider* provider);

[VB] Function ToByte(ByVal provider As IFormatProvider) As Byte Implements

```

1 IConvertible.ToByte
2 [JScript] function IConvertible.ToByte(provider : IFormatProvider) : Byte;
3     IConvertible.ToChar
4
5 [C#] char IConvertible.ToChar(IFormatProvider provider);
6 [C++] __wchar_t IConvertible::ToChar(IFormatProvider* provider);
7 [VB] Function ToChar(ByVal provider As IFormatProvider) As Char Implements
8 IConvertible.ToChar
9 [JScript] function IConvertible.ToChar(provider : IFormatProvider) : Char;
10     IConvertible.ToDateTime
11
12 [C#] DateTime IConvertible.ToDateTime(IFormatProvider provider);
13 [C++] DateTime IConvertible::ToDateTime(IFormatProvider* provider);
14 [VB] Function ToDateTime(ByVal provider As IFormatProvider) As DateTime
15 Implements IConvertible.ToDateTime
16 [JScript] function IConvertible.ToDateTime(provider : IFormatProvider) :
17 DateTime;
18     IConvertible.ToDecimal
19
20 [C#] decimal IConvertible.ToDecimal(IFormatProvider provider);
21 [C++] Decimal IConvertible::ToDecimal(IFormatProvider* provider);
22 [VB] Function ToDecimal(ByVal provider As IFormatProvider) As Decimal
23 Implements IConvertible.ToDecimal
24 [JScript] function IConvertible.ToDecimal(provider : IFormatProvider) : Decimal;
25     IConvertible.ToDouble

```

```

1
2 [C#] double IConvertible.ToDouble(IFormatProvider provider);
3 [C++] double IConvertible::ToDouble(IFormatProvider* provider);
4 [VB] Function ToDouble(ByVal provider As IFormatProvider) As Double
5 Implements IConvertible.ToDouble
6 [JScript] function IConvertible.ToDouble(provider : IFormatProvider) : double;
7     IConvertible.ToInt16
8
9 [C#] short IConvertible.ToInt16(IFormatProvider provider);
10 [C++] short IConvertible::ToInt16(IFormatProvider* provider);
11 [VB] Function ToInt16(ByVal provider As IFormatProvider) As Short
12 Implements IConvertible.ToInt16
13 [JScript] function IConvertible.ToInt16(provider : IFormatProvider) : Int16;
14     IConvertible.ToInt32
15
16 [C#] int IConvertible.ToInt32(IFormatProvider provider);
17 [C++] int IConvertible::ToInt32(IFormatProvider* provider);
18 [VB] Function ToInt32(ByVal provider As IFormatProvider) As Integer
19 Implements IConvertible.ToInt32
20 [JScript] function IConvertible.ToInt32(provider : IFormatProvider) : int;
21     IConvertible.ToInt64
22
23 [C#] long IConvertible.ToInt64(IFormatProvider provider);
24 [C++] __int64 IConvertible::ToInt64(IFormatProvider* provider);
25 [VB] Function ToInt64(ByVal provider As IFormatProvider) As Long Implements

```



```

1  IConvertible.ToInt64
2  [JScript] function IConvertible.ToInt64(provider : IFormatProvider) : long;
3      IConvertible.ToSByte
4
5  [C#] sbyte IConvertible.ToSByte(IFormatProvider provider);
6  [C++] char IConvertible::ToSByte(IFormatProvider* provider);
7  [VB] Function ToSByte(ByVal provider As IFormatProvider) As SByte
8  Implements IConvertible.ToSByte
9  [JScript] function IConvertible.ToSByte(provider : IFormatProvider) : SByte;
10     IConvertible.ToSingle
11
12  [C#] float IConvertible.ToSingle(IFormatProvider provider);
13  [C++] float IConvertible::ToSingle(IFormatProvider* provider);
14  [VB] Function ToSingle(ByVal provider As IFormatProvider) As Single
15  Implements IConvertible.ToSingle
16  [JScript] function IConvertible.ToSingle(provider : IFormatProvider) : float;
17     IConvertible.ToType
18
19  [C#] object IConvertible.ToType(Type type, IFormatProvider provider);
20  [C++] Object* IConvertible::ToType(Type* type, IFormatProvider* provider);
21  [VB] Function ToType(ByVal type As Type, ByVal provider As IFormatProvider)
22  As Object Implements IConvertible.ToType
23  [JScript] function IConvertible.ToType(type : Type, provider : IFormatProvider) :
24  Object;
25     IConvertible.ToUInt16

```

```

1
2 [C#] ushort IConvertible.ToUInt16(IFormatProvider provider);
3 [C++] unsigned short IConvertible::ToUInt16(IFormatProvider* provider);
4 [VB] Function ToUInt16(ByVal provider As IFormatProvider) As UInt16
5 Implements IConvertible.ToUInt16
6 [JScript] function IConvertible.ToUInt16(provider : IFormatProvider) : UInt16;
7     IConvertible.ToUInt32
8
9 [C#] uint IConvertible.ToUInt32(IFormatProvider provider);
10 [C++] unsigned int IConvertible::ToUInt32(IFormatProvider* provider);
11 [VB] Function ToUInt32(ByVal provider As IFormatProvider) As UInt32
12 Implements IConvertible.ToUInt32
13 [JScript] function IConvertible.ToUInt32(provider : IFormatProvider) : UInt32;
14     IConvertible.ToUInt64
15
16 [C#] ulong IConvertible.ToUInt64(IFormatProvider provider);
17 [C++] unsigned __int64 IConvertible::ToUInt64(IFormatProvider* provider);
18 [VB] Function ToUInt64(ByVal provider As IFormatProvider) As UInt64
19 Implements IConvertible.ToUInt64
20 [JScript] function IConvertible.ToUInt64(provider : IFormatProvider) : UInt64;
21     ToString
22
23 [C#] public override string ToString();
24 [C++] public: String* ToString();
25 [VB] Overrides Public Function ToString() As String

```

[JScript] public override function ToString() : String; Converts the numeric value of this instance to its equivalent **String** representation.

Description

Converts the numeric value of this instance to its equivalent **String** representation.

Return Value: The **System.String** representation of the value of this instance, consisting of a sequence of digits ranging from 0 to 9, without a sign or leading zeroes.

The return value is formatted with the general format specifier ("G") and the **System.Globalization.NumberFormatInfo** for the current culture.

ToString

[C#] public string ToString(IFormatProvider provider);

[C++] public: __sealed String* ToString(IFormatProvider* provider);

[VB] NotOverridable Public Function ToString(ByVal provider As IFormatProvider) As String

[JScript] public function ToString(provider : IFormatProvider) : String;

Description

Converts the numeric value of this instance to its equivalent **String** representation using the specified culture-specific format information.

Return Value: The **System.String** representation of the value of this instance as specified by *provider*.

This instance is formatted with the general format specifier ("G"). An **System.IFormatProvider** interface implementation which supplies culture-specific formatting information.

ToString

```
[C#] public string ToString(string format);  
[C++] public: String* ToString(String* format);  
[VB] Public Function ToString(ByVal format As String) As String  
[JScript] public function ToString(format : String) : String;
```

Description

Converts the numeric value of this instance to its equivalent **String** representation using the specified format.

Return Value: The **System.String** representation of the value of this instance as specified by *format* .

If *format* is **null** or an empty string ("") the return value of this instance is formatted with the general format specifier ("G"). A format string.

ToString

```
[C#] public string ToString(string format, IFormatProvider provider);  
[C++] public: __sealed String* ToString(String* format, IFormatProvider*  
provider);  
[VB] NotOverridable Public Function ToString(ByVal format As String, ByVal  
provider As IFormatProvider) As String  
[JScript] public function ToString(format : String, provider : IFormatProvider) :
```

String;

Description

Converts the numeric value of this instance to its equivalent **String** representation using the specified format and culture-specific format information.

Return Value: The **System.String** representation of the value of this instance as specified by *format* and *provider* .

If *format* is **null** or an empty string ("") the return value for this instance is formatted with the general format specifier ("G"). A format specification. An **System.IFormatProvider** interface implementation which supplies culture-specific formatting information about this instance.

UInt64 structure (System)

ToString

Description

Represents a 64-bit unsigned integer.

The **UInt64** value type represents unsigned integers with values ranging from 0 to 184,467,440,737,095,551,615.

ToString

[C#] public const ulong MaxValue;

[C++] public: const unsigned __int64 MaxValue;

[VB] Public Const MaxValue As UInt64

[JScript] public var MaxValue : UInt64;

1
2 *Description*

3 A constant representing the largest possible value of **UInt64** .

4 The value of this constant is 18,446,744,073,709,551,615; that is,
5 hexadecimal 0xFFFFFFFFFFFFFFFF.

6 *ToString*

7
8 [C#] public const ulong MinValue;

9 [C++] public: const unsigned __int64 MinValue;

10 [VB] Public Const MinValue As UInt64

11 [JScript] public var MinValue : UInt64;

12
13 *Description*

14 A constant representing the smallest possible value of **UInt64** .

15 The value of this constant is 0.

16 *CompareTo*

17
18 [C#] public int CompareTo(object value);

19 [C++] public: __sealed int CompareTo(Object* value);

20 [VB] NotOverridable Public Function CompareTo(ByVal value As Object) As

21 *Integer*

22 [JScript] public function CompareTo(value : Object) : int;

23
24 *Description*

1 Compares this instance to a specified object and returns an indication of
2 their relative values.

3 *Return Value:* A signed number indicating the relative values of this instance and
4 *value* .

5 Any instance of **UInt64** , regardless of its value, is considered greater than
6 **null** . An object to compare, or **null**.

7 Equals

9 [C#] public override bool Equals(object obj);

10 [C++] public: bool Equals(Object* obj);

11 [VB] Overrides Public Function Equals(ByVal obj As Object) As Boolean

12 [JScript] public override function Equals(obj : Object) : Boolean;

14 *Description*

15 Returns a value indicating whether this instance is equal to a specified
16 object.

17 *Return Value:* **true** if *obj* is an instance of **UInt64** and equals the value of this
18 instance; otherwise, **false** . An object to compare with this instance.

19 GetHashCode

21 [C#] public override int GetHashCode();

22 [C++] public: int GetHashCode();

23 [VB] Overrides Public Function GetHashCode() As Integer

24 [JScript] public override function GetHashCode() : int;

1
2 *Description*

3 Returns the hash code for this instance.

4 *Return Value:* A 32-bit signed integer hash code.

5 **GetTypeCode**

6
7 [C#] public TypeCode GetTypeCode();

8 [C++] public: __sealed TypeCode GetTypeCode();

9 [VB] NotOverridable Public Function GetTypeCode() As TypeCode

10 [JScript] public function GetTypeCode() : TypeCode;

11
12 *Description*

13 Returns the **TypeCode** for value type **UInt64** .

14 *Return Value:* The enumerated constant, **System.TypeCode.UInt64** .

15 **Parse**

16
17 [C#] public static ulong Parse(string s);

18 [C++] public: static unsigned __int64 Parse(String* s);

19 [VB] Public Shared Function Parse(ByVal s As String) As UInt64

20 [JScript] public static function Parse(s : String) : UInt64; Converts the **String**
21 representation of a number to its 64-bit unsigned integer equivalent.

22
23 *Description*

24

25

Converts the **String** representation of a number to its 64-bit unsigned integer equivalent.

Return Value: An 64-bit unsigned integer equivalent to the number contained in *s* .

s contains a number of the form: [ws][sign]digits[ws] Items in square brackets ('[' and ']') are optional, and other items are as follows. A **System.String** containing a number to convert.

Parse

```
[C#] public static ulong Parse(string s, IFormatProvider provider);  
[C++] public: static unsigned __int64 Parse(String* s, IFormatProvider*  
provider);  
[VB] Public Shared Function Parse(ByVal s As String, ByVal provider As  
IFormatProvider) As UInt64  
[JScript] public static function Parse(s : String, provider : IFormatProvider) :  
UInt64;
```

Description

Converts the **String** representation of a number in a specified culture-specific format to its 64-bit unsigned integer equivalent.

Return Value: An 64-bit unsigned integer equivalent to the number specified in *s* .

s contains a number of the form: [ws][sign]digits[ws] Items in square brackets ('[' and ']') are optional, and other items are as follows. A **System.String** containing a number to convert. An **System.IFormatProvider** interface implementation which supplies culture-specific formatting information about *s*.

Parse

```

1
2 [C#] public static ulong Parse(string s, NumberStyles style);
3 [C++] public: static unsigned __int64 Parse(String* s, NumberStyles style);
4 [VB] Public Shared Function Parse(ByVal s As String, ByVal style As
5 NumberStyles) As UInt64
6 [JScript] public static function Parse(s : String, style : NumberStyles) : UInt64;

```

Description

Converts the **String** representation of a number in a specified style to its 64-bit unsigned integer equivalent.

Return Value: An 64-bit unsigned integer equivalent to the number specified in *s*.

s contains a number of the form: [ws][sign]digits[ws] Items in square brackets ('[' and ']') are optional, and other items are as follows. A **System.String** containing a number to convert. The combination of one or more **System.Globalization.NumberStyles** constants that indicate the permitted format of *s*.

Parse

```

18
19 [C#] public static ulong Parse(string s, NumberStyles style, IFormatProvider
20 provider);
21 [C++] public: static unsigned __int64 Parse(String* s, NumberStyles style,
22 IFormatProvider* provider);
23 [VB] Public Shared Function Parse(ByVal s As String, ByVal style As
24 NumberStyles, ByVal provider As IFormatProvider) As UInt64
25 [JScript] public static function Parse(s : String, style : NumberStyles, provider :

```

1 IFormatProvider) : UInt64;

3 *Description*

4 Converts the **String** representation of a number in a specified style and
5 culture-specific format to its 64-bit unsigned integer equivalent.

6 *Return Value:* An 64-bit unsigned integer equivalent to the number specified in *s* .

7 *s* contains a number of the form: [ws][sign]digits[ws] Items in square
8 brackets ('[' and ']') are optional, and other items are as follows. A **System.String**
9 containing a number to convert. The combination of one or more
10 **System.Globalization.NumberStyles** constants that indicate the permitted format
11 of *s*. An **System.IFormatProvider** interface implementation which supplies
12 culture-specific formatting information about *s*.

13 IConvertible.ToBoolean

15 [C#] bool IConvertible.ToBoolean(IFormatProvider provider);

16 [C++] bool IConvertible::ToBoolean(IFormatProvider* provider);

17 [VB] Function ToBoolean(ByVal provider As IFormatProvider) As Boolean

18 Implements IConvertible.ToBoolean

19 [JScript] function IConvertible.ToBoolean(provider : IFormatProvider) : Boolean;

20 IConvertible.ToByte

22 [C#] byte IConvertible.ToByte(IFormatProvider provider);

23 [C++] unsigned char IConvertible::ToByte(IFormatProvider* provider);

24 [VB] Function ToByte(ByVal provider As IFormatProvider) As Byte Implements

```

1  IConvertible.ToByte
2  [JScript] function IConvertible.ToByte(provider : IFormatProvider) : Byte;
3
4  IConvertible.ToChar
5
6  [C#] char IConvertible.ToChar(IFormatProvider provider);
7  [C++] __wchar_t IConvertible::ToChar(IFormatProvider* provider);
8  [VB] Function ToChar(ByVal provider As IFormatProvider) As Char Implements
9  IConvertible.ToChar
10 [JScript] function IConvertible.ToChar(provider : IFormatProvider) : Char;
11
12 IConvertible.ToDateTime
13
14 [C#] DateTime IConvertible.ToDateTime(IFormatProvider provider);
15 [C++] DateTime IConvertible::ToDateTime(IFormatProvider* provider);
16 [VB] Function ToDateTime(ByVal provider As IFormatProvider) As DateTime
17 Implements IConvertible.ToDateTime
18 [JScript] function IConvertible.ToDateTime(provider : IFormatProvider) :
19 DateTime;
20
21 IConvertible.ToDecimal
22
23 [C#] decimal IConvertible.ToDecimal(IFormatProvider provider);
24 [C++] Decimal IConvertible::ToDecimal(IFormatProvider* provider);
25 [VB] Function ToDecimal(ByVal provider As IFormatProvider) As Decimal
26 Implements IConvertible.ToDecimal
27 [JScript] function IConvertible.ToDecimal(provider : IFormatProvider) : Decimal;
28
29 IConvertible.ToDouble

```

```

1
2 [C#] double IConvertible.ToDouble(IFormatProvider provider);
3 [C++] double IConvertible::ToDouble(IFormatProvider* provider);
4 [VB] Function ToDouble(ByVal provider As IFormatProvider) As Double
5 Implements IConvertible.ToDouble
6 [JScript] function IConvertible.ToDouble(provider : IFormatProvider) : double;
7     IConvertible.ToInt16
8
9 [C#] short IConvertible.ToInt16(IFormatProvider provider);
10 [C++] short IConvertible::ToInt16(IFormatProvider* provider);
11 [VB] Function ToInt16(ByVal provider As IFormatProvider) As Short
12 Implements IConvertible.ToInt16
13 [JScript] function IConvertible.ToInt16(provider : IFormatProvider) : Int16;
14     IConvertible.ToInt32
15
16 [C#] int IConvertible.ToInt32(IFormatProvider provider);
17 [C++] int IConvertible::ToInt32(IFormatProvider* provider);
18 [VB] Function ToInt32(ByVal provider As IFormatProvider) As Integer
19 Implements IConvertible.ToInt32
20 [JScript] function IConvertible.ToInt32(provider : IFormatProvider) : int;
21     IConvertible.ToInt64
22
23 [C#] long IConvertible.ToInt64(IFormatProvider provider);
24 [C++] __int64 IConvertible::ToInt64(IFormatProvider* provider);
25 [VB] Function ToInt64(ByVal provider As IFormatProvider) As Long Implements

```

```

1  IConvertible.ToInt64
2  [JScript] function IConvertible.ToInt64(provider : IFormatProvider) : long;
3      IConvertible.ToSByte
4
5  [C#] sbyte IConvertible.ToSByte(IFormatProvider provider);
6  [C++] char IConvertible::ToSByte(IFormatProvider* provider);
7  [VB] Function ToSByte(ByVal provider As IFormatProvider) As SByte
8  Implements IConvertible.ToSByte
9  [JScript] function IConvertible.ToSByte(provider : IFormatProvider) : SByte;
10     IConvertible.ToSingle
11
12  [C#] float IConvertible.ToSingle(IFormatProvider provider);
13  [C++] float IConvertible::ToSingle(IFormatProvider* provider);
14  [VB] Function ToSingle(ByVal provider As IFormatProvider) As Single
15  Implements IConvertible.ToSingle
16  [JScript] function IConvertible.ToSingle(provider : IFormatProvider) : float;
17     IConvertible.ToType
18
19  [C#] object IConvertible.ToType(Type type, IFormatProvider provider);
20  [C++] Object* IConvertible::ToType(Type* type, IFormatProvider* provider);
21  [VB] Function ToType(ByVal type As Type, ByVal provider As IFormatProvider)
22  As Object Implements IConvertible.ToType
23  [JScript] function IConvertible.ToType(type : Type, provider : IFormatProvider) :
24  Object;
25     IConvertible.ToUInt16

```

```

1
2 [C#] ushort IConvertible.ToUInt16(IFormatProvider provider);
3 [C++] unsigned short IConvertible::ToUInt16(IFormatProvider* provider);
4 [VB] Function ToUInt16(ByVal provider As IFormatProvider) As UInt16
5 Implements IConvertible.ToUInt16
6 [JScript] function IConvertible.ToUInt16(provider : IFormatProvider) : UInt16;
7     IConvertible.ToUInt32
8
9 [C#] uint IConvertible.ToUInt32(IFormatProvider provider);
10 [C++] unsigned int IConvertible::ToUInt32(IFormatProvider* provider);
11 [VB] Function ToUInt32(ByVal provider As IFormatProvider) As UInt32
12 Implements IConvertible.ToUInt32
13 [JScript] function IConvertible.ToUInt32(provider : IFormatProvider) : UInt32;
14     IConvertible.ToUInt64
15
16 [C#] ulong IConvertible.ToUInt64(IFormatProvider provider);
17 [C++] unsigned __int64 IConvertible::ToUInt64(IFormatProvider* provider);
18 [VB] Function ToUInt64(ByVal provider As IFormatProvider) As UInt64
19 Implements IConvertible.ToUInt64
20 [JScript] function IConvertible.ToUInt64(provider : IFormatProvider) : UInt64;
21     ToString
22
23 [C#] public override string ToString();
24 [C++] public: String* ToString();
25 [VB] Overrides Public Function ToString() As String

```

[JScript] public override function ToString() : String; Converts the numeric value of this instance to its equivalent **String** representation.

Description

Converts the numeric value of this instance to its equivalent **String** representation.

Return Value: The **System.String** representation of the value of this instance, consisting of a sequence of digits ranging from 0 to 9, without a sign or leading zeroes.

The return value is formatted with the general format specifier ("G") and the **System.Globalization.NumberFormatInfo** for the current culture.

ToString

[C#] public string ToString(IFormatProvider provider);

[C++] public: __sealed String* ToString(IFormatProvider* provider);

[VB] NotOverridable Public Function ToString(ByVal provider As IFormatProvider) As String

[JScript] public function ToString(provider : IFormatProvider) : String;

Description

Converts the numeric value of this instance to its equivalent **String** representation using the specified culture-specific format information.

Return Value: The **System.String** representation of the value of this instance as specified by *provider*.

This instance is formatted with the general format specifier ("G"). An **System.IFormatProvider** interface implementation which supplies culture-specific formatting information.

ToString

```
[C#] public string ToString(string format);  
[C++] public: String* ToString(String* format);  
[VB] Public Function ToString(ByVal format As String) As String  
[JScript] public function ToString(format : String) : String;
```

Description

Converts the numeric value of this instance to its equivalent **String** representation using the specified format.

Return Value: The **System.String** representation of the value of this instance as specified by *format*.

If *format* is **null** or an empty string (""), the return value of this instance is formatted with the general format specifier ("G"). A format string.

ToString

```
[C#] public string ToString(string format, IFormatProvider provider);  
[C++] public: __sealed String* ToString(String* format, IFormatProvider*  
provider);  
[VB] NotOverridable Public Function ToString(ByVal format As String, ByVal  
provider As IFormatProvider) As String  
[JScript] public function ToString(format : String, provider : IFormatProvider) :
```

String;

Description

Converts the numeric value of this instance to its equivalent **String** representation using the specified format and culture-specific format information.

Return Value: The **System.String** representation of the value of this instance as specified by *format* and *provider* .

If *format* is **null** or an empty string (""), the return value for this instance is formatted with the general format specifier ("G"). A format specification. An **System.IFormatProvider** interface implementation which supplies culture-specific formatting information about this instance.

UIntPtr structure (System)

ToString

Description

A platform-specific type that is used to represent a pointer or a handle.

The **System.UIntPtr** type is designed to be a platform-specific, machine-sized integer. That is, an instance of this type is expected to be 32-bits on 32-bit hardware and operating systems, and 64-bits on 64-bit hardware and operating systems.

ToString

[C#] public static readonly UIntPtr Zero;

[C++] public: static UIntPtr Zero;

1 [VB] Public Shared ReadOnly Zero As UIntPtr

2 [JScript] public static var Zero : UIntPtr;

3
4 *Description*

5 A read-only field that represents an uninitialized pointer or handle.

6 The value of this field is not equivalent to **null** , but is instead a pointer
7 which has not been assigned any value whatsoever. Use this field to efficiently
8 determine whether an instance of **UIntPtr** has been set.

9 UIntPtr

10 *Example Syntax:*

11 ToString

12
13 [C#] public UIntPtr(uint value);

14 [C++] public: UIntPtr(unsigned int value);

15 [VB] Public Sub New(ByVal value As UInt32)

16 [JScript] public function UIntPtr(value : UInt32); Initializes a new instance of the
17 **System.UIntPtr** structure.

18
19 *Description*

20 Initializes a new instance of the **System.UIntPtr** structure to the specified
21 32-bit pointer or handle. A pointer or handle contained in a 32-bit unsigned
22 integer.

23 UIntPtr

24 *Example Syntax:*

25 ToString

```

1
2 [C#] public UIntPtr(ulong value);
3 [C++] public: UIntPtr(unsigned __int64 value);
4 [VB] Public Sub New(ByVal value As UInt64)
5 [JScript] public function UIntPtr(value : UInt64);
6

```

7 *Description*

8 Initializes a new instance of the **System.UIntPtr** structure to the specified
9 64-bit pointer or handle.

10 An exception is only thrown if the value of *value* requires more bits than
11 the current platform supports. A pointer or handle contained in a 64-bit unsigned
12 integer.

13 UIntPtr

14 *Example Syntax:*

15 ToString

16
17 [C#] unsafe public UIntPtr(void* value);

18 [C++] public: UIntPtr(void* value);

19 Size

20 ToString

21
22 [C#] public static int Size {get;}

23 [C++] public: __property static int get_Size();

24 [VB] Public Shared ReadOnly Property Size As Integer

25 [JScript] public static function get Size() : int;

1
2 *Description*

3 Gets the size of this instance.

4 Equals

5
6 [C#] public override bool Equals(object obj);

7 [C++] public: bool Equals(Object* obj);

8 [VB] Overrides Public Function Equals(ByVal obj As Object) As Boolean

9 [JScript] public override function Equals(obj : Object) : Boolean;

10
11 *Description*

12 Returns a value indicating whether this instance is equal to a specified
13 object.

14 *Return Value:* **true** if *obj* is an instance of **UIntPtr** and equals the value of this
15 instance; otherwise, **false** . An object to compare with this instance or **null**.

16 GetHashCode

17
18 [C#] public override int GetHashCode();

19 [C++] public: int GetHashCode();

20 [VB] Overrides Public Function GetHashCode() As Integer

21 [JScript] public override function GetHashCode() : int;

22
23 *Description*

24 Returns the hash code for this instance.

25 *Return Value:* A 32-bit signed integer hash code.

1 op_Equality

2
3 [C#] public static bool operator ==(UIntPtr value1, UIntPtr value2);
4 [C++] public: static bool op_Equality(UIntPtr value1, UIntPtr value2);
5 [VB] returnValue = UIntPtr.op_Equality(value1, value2)
6 [JScript] returnValue = value1 == value2;

7
8 *Description*

9 Determines whether two specified instances of **System.UIntPtr** are equal.

10 *Return Value:* **true** if *value1* equals *value2* ; otherwise, **false** . A **UIntPtr**. A

11 **UIntPtr.**

12 op_Explicit

13
14 [C#] public static explicit operator UIntPtr(uint value);
15 [C++] public: static UIntPtr op_Explicit(unsigned int value);
16 [VB] returnValue = UIntPtr.op_Explicit(value)
17 [JScript] returnValue = UIntPtr(value);

18
19 *Description*

20 Converts the value of a 32-bit unsigned integer to an **System.UIntPtr** .

21 *Return Value:* A new instance of **System.UIntPtr** initialized to *value* . A 32-bit
22 unsigned integer.

23 op_Explicit

24
25 [C#] public static explicit operator UIntPtr(ulong value);

1 [C++] public: static UIntPtr op_Explicit(unsigned __int64 value);

2 [VB] returnValue = UIntPtr.op_Explicit(value)

3 [JScript] returnValue = UIntPtr(value);

4
5 *Description*

6 Converts the value of a 64-bit unsigned integer to an **System.UIntPtr** .

7 *Return Value:* A new instance of **System.UIntPtr** initialized to *value* . A 64-bit
8 unsigned integer.

9 op_Explicit

10
11 [C#] unsafe public static explicit operator void*(UIntPtr value);

12 [C++] public: static void* op_Explicit();

13 op_Explicit

14
15 [C#] public static explicit operator ulong(UIntPtr value);

16 [C++] public: static unsigned __int64 op_Explicit();

17 [VB] returnValue = UIntPtr.op_Explicit(value)

18 [JScript] returnValue = UInt64(value);

19
20 *Description*

21 Converts the value of the specified **System.UIntPtr** instance to a 64-bit
22 unsigned integer. A **UIntPtr**.

23 op_Explicit

24
25 [C#] public static explicit operator uint(UIntPtr value);

1 [C++] public: static unsigned int op_Explicit();

2 [VB] returnValue = UIntPtr.op_Explicit(value)

3 [JScript] returnValue = UInt32(value);

4
5 *Description*

6 Converts the value of the specified **System.UIntPtr** instance to a 32-bit
7 unsigned integer.

8 An exception is only thrown if the value of *value* requires more bits than
9 the current platform supports. A **UIntPtr**.

10 op_Explicit

11
12 [C#] unsafe public static explicit operator UIntPtr(void* value);

13 [C++] public: static UIntPtr op_Explicit(void* value);

14 op_Inequality

15
16 [C#] public static bool operator !=(UIntPtr value1, UIntPtr value2);

17 [C++] public: static bool op_Inequality(UIntPtr value1, UIntPtr value2);

18 [VB] returnValue = UIntPtr.op_Inequality(value1, value2)

19 [JScript] returnValue = value1 != value2;

20
21 *Description*

22 Determines whether two specified instances of **System.UIntPtr** are not
23 equal.

24 *Return Value:* **true** if *value1* does not equal *value2* ; otherwise, **false** . A **UIntPtr**.

25 A **UIntPtr**.

ISerializable.GetObjectData

[C#] void ISerializable.GetObjectData(SerializationInfo info, StreamingContext context);

[C++] void ISerializable::GetObjectData(SerializationInfo* info, StreamingContext context);

[VB] Sub GetObjectData(ByVal info As SerializationInfo, ByVal context As StreamingContext) Implements ISerializable.GetObjectData

[JScript] function ISerializable.GetObjectData(info : SerializationInfo, context : StreamingContext);

ToPointer

[C#] unsafe public void* ToPointer();

[C++] public: void* ToPointer();

Description

Converts the value of this instance to a pointer to an unspecified type.

Return Value: A pointer to **System.Void** ; that is, a pointer to memory containing data of an unspecified type.

ToString

[C#] public override string ToString();

[C++] public: String* ToString();

[VB] Overrides Public Function ToString() As String

[JScript] public override function ToString() : String;

1
2 *Description*

3 Converts the numeric value of this instance to its equivalent **String**
4 representation.

5 *Return Value:* The **System.String** representation of the value of this instance.

6 ToUInt32

7
8 [C#] public uint ToUInt32();

9 [C++] public: unsigned int ToUInt32();

10 [VB] Public Function ToUInt32() As UInt32

11 [JScript] public function ToUInt32() : UInt32;

12
13 *Description*

14 Converts the value of this instance to a 32-bit unsigned integer.

15 *Return Value:* A 32-bit unsigned integer.

16 An exception is only thrown if the value of *value* requires more bits than
17 the current platform supports.

18 ToUInt64

19
20 [C#] public ulong ToUInt64();

21 [C++] public: unsigned __int64 ToUInt64();

22 [VB] Public Function ToUInt64() As UInt64

23 [JScript] public function ToUInt64() : UInt64;

24
25 *Description*

Converts the value of this instance to a 64-bit unsigned integer.

Return Value: A 64-bit unsigned integer.

UnauthorizedAccessException class (System)

ToUInt64

Description

The exception that is thrown when the operating system denies access because of an I/O error or a specific type of security error.

UnauthorizedAccessException uses the HRESULT COR_E_UNAUTHORIZEDACCESS, which has the value 0x80070005.

UnauthorizedAccessException

Example Syntax:

ToUInt64

[C#] public UnauthorizedAccessException();

[C++] public: UnauthorizedAccessException();

[VB] Public Sub New()

[JScript] public function UnauthorizedAccessException(); Initializes a new instance of the **System.UnauthorizedAccessException** class.

Description

Initializes a new instance of the **System.UnauthorizedAccessException** class.

UnauthorizedAccessException

Example Syntax:

ToUInt64

[C#] public UnauthorizedAccessException(string message);

[C++] public: UnauthorizedAccessException(String* message);

[VB] Public Sub New(ByVal message As String)

[JScript] public function UnauthorizedAccessException(message : String);

Description

Initializes a new instance of the **System.UnauthorizedAccessException** class with a specified error message. The error message that explains the reason for the exception.

UnauthorizedAccessException

Example Syntax:

ToUInt64

[C#] protected UnauthorizedAccessException(SerializationInfo info,
StreamingContext context);

[C++] protected: UnauthorizedAccessException(SerializationInfo* info,
StreamingContext context);

[VB] Protected Sub New(ByVal info As SerializationInfo, ByVal context As
StreamingContext)

[JScript] protected function UnauthorizedAccessException(info :
SerializationInfo, context : StreamingContext);

Description

Initializes a new instance of the **System.UnauthorizedAccessException** class with serialized data. The **System.Runtime.Serialization.SerializationInfo** that holds the serialized object data about the exception being thrown. The **System.Runtime.Serialization.StreamingContext** that contains contextual information about the source or destination.

UnauthorizedAccessException

Example Syntax:

ToUInt64

```
[C#] public UnauthorizedAccessException(string message, Exception inner);  
[C++] public: UnauthorizedAccessException(String* message, Exception* inner);  
[VB] Public Sub New(ByVal message As String, ByVal inner As Exception)  
[JScript] public function UnauthorizedAccessException(message : String, inner :  
Exception);
```

Description

Initializes a new instance of the **System.UnauthorizedAccessException** class with a specified error message and a reference to the inner exception that is the root cause of this exception.

When an **Exception** *X* is thrown as a direct result of a previous exception *Y*, the **System.Exception.InnerException** property of *X* should contain a reference to *Y*. The **InnerException** property returns the same value as was passed into the constructor, or **null** if the inner exception value was not supplied to the

1 constructor. The error message that explains the reason for the exception. An
2 instance of **System.Exception** that is the cause of the current **Exception**. If the
3 *inner* parameter is non-null, then the current **Exception** is raised in a catch block
4 handling *inner* .

5 HelpLink

6 HResult

7 InnerException

8 Message

9 Source

10 StackTrace

11 TargetSite

12 UnhandledExceptionEventArgs class (System)

13 ToString

14
15
16 *Description*

17 Provides data for the event that is raised when there is an exception that is
18 not handled by the application domain.

19 **System.UnhandledExceptionEventArgs** provides access to the exception
20 object and a flag indicating whether the common language runtime is terminating.
21 The **System.UnhandledExceptionEventArgs** is one of the parameters passed into
22 **System.UnhandledExceptionHandler** .

23 UnhandledExceptionEventArgs

24 *Example Syntax:*

25 ToString

```

1
2 [C#] public UnhandledExceptionEventArgs(object exception, bool isTerminating);
3 [C++] public: UnhandledExceptionEventArgs(Object* exception, bool
4 isTerminating);
5 [VB] Public Sub New(ByVal exception As Object, ByVal isTerminating As
6 Boolean)
7 [JScript] public function UnhandledExceptionEventArgs(exception : Object,
8 isTerminating : Boolean);
9

```

Description

Initializes a new instance of the **System.UnhandledExceptionEventArgs** class with the exception object and a common language runtime termination flag. The exception that is not handled. **true** if the runtime is terminating; otherwise, **false**.

ExceptionObject

ToString

```

17
18 [C#] public object ExceptionObject {get;}
19 [C++] public: __property Object* get_ExceptionObject();
20 [VB] Public ReadOnly Property ExceptionObject As Object
21 [JScript] public function get ExceptionObject() : Object;
22

```

Description

Gets the unhandled exception object.

IsTerminating

ToString

[C#] public bool IsTerminating {get;}

[C++] public: __property bool get_IsTerminating();

[VB] Public ReadOnly Property IsTerminating As Boolean

[JScript] public function get IsTerminating() : Boolean;

Description

Indicates whether the common language runtime is terminating.

UnhandledExceptionEventHandler delegate (System)

ToString

Description

Represents the method that will handle the event triggered by an exception that is not handled by the application domain. The source of the unhandled exception event. An *UnhandledExceptionEventArgs* that contains the event data.

When you create an **System.UnhandledExceptionEventHandler** delegate, you identify the method that will handle the event. To associate the event handler with your event, add an instance of the delegate to the event. The event handler is called whenever the event occurs, unless you remove the delegate. For more information about event handler delegates, see .

ValueType class (System)

ToString

1
2
3 *Description*

4 Provides the base class for value types.

5 **ValueType** overrides the virtual methods from **System.Object** with more
6 appropriate implementations for value types. See also **System.Enum** , which
7 inherits from **ValueType** .

8 **ValueType**

9 *Example Syntax:*

10 **ToString**

11
12 [C#] protected ValueType();

13 [C++] protected: ValueType();

14 [VB] Protected Sub New()

15 [JScript] protected function ValueType();

16 **Equals**

17
18 [C#] public override bool Equals(object obj);

19 [C++] public: bool Equals(Object* obj);

20 [VB] Overrides Public Function Equals(ByVal obj As Object) As Boolean

21 [JScript] public override function Equals(obj : Object) : Boolean;

22
23 *Description*
24
25

Indicates whether this instance and a specified object are equal.

Return Value: **true** if *obj* and this instance are the same type and represent the same value; otherwise, **false** . Another object to compare to.

GetHashCode

[C#] public override int GetHashCode();

[C++] public: int GetHashCode();

[VB] Overrides Public Function GetHashCode() As Integer

[JScript] public override function GetHashCode() : int;

Description

Returns the hash code for this instance.

Return Value: A 32-bit signed integer that is the hash code for this instance.

Any value types that are inserted into a Hashtable must override GetHashCode() for performance reasons. The default implementation on ValueType will return the same value for all instances of a particular type, which makes hashing as slow as a linked list. See the documentation for Object's GetHashCode for more details.

ToString

[C#] public override string ToString();

[C++] public: String* ToString();

[VB] Overrides Public Function ToString() As String

[JScript] public override function ToString() : String;

1
2 *Description*

3 Returns the fully qualified type name of this instance.

4 *Return Value:* A **System.String** containing a fully qualified type name.

5 Version class (System)

6 ToString

7
8
9 *Description*

10 Represents the version number for a common language runtime assembly.

11 This class cannot be inherited.

12 Version numbers consist of four components: *major* , *minor* , *build* , and
13 *revision* . Components *major* and *minor* are required. Component *revision* is only
14 optional if *build* is not defined.

15 Version

16 *Example Syntax:*

17 ToString

18
19 [C#] public Version();

20 [C++] public: Version();

21 [VB] Public Sub New()

22 [JScript] public function Version();

23
24 *Description*

25 Initializes a new instance of the **System.Version** class.

Version

Example Syntax:

ToString

[C#] public Version(string version);

[C++] public: Version(String* version);

[VB] Public Sub New(ByVal version As String)

[JScript] public function Version(version : String);

Description

Initializes a new instance of the **Version** class using the value represented by the specified **String** .

version can only contain components *major* , *minor* , *build* , and *revision*, in that order and all separated by periods. There must be at least two components, and at most four. The first two components are assumed to be *major* and *minor* . A string containing the major, minor, build, and revision numbers, where each number is delimited with a period character ('.').

Version

Example Syntax:

ToString

[C#] public Version(int major, int minor);

[C++] public: Version(int major, int minor);

[VB] Public Sub New(ByVal major As Integer, ByVal minor As Integer)

[JScript] public function Version(major : int, minor : int);

Description

Initializes a new instance of the **Version** class using the specified major and minor values.

Metadata restricts *major* and *minor* to a maximum value of **System.UInt16.MaxValue** - 1. The major version number. The minor version number.

Version

Example Syntax:

ToString

```
[C#] public Version(int major, int minor, int build);
```

```
[C++] public: Version(int major, int minor, int build);
```

```
[VB] Public Sub New(ByVal major As Integer, ByVal minor As Integer, ByVal  
build As Integer)
```

```
[JScript] public function Version(major : int, minor : int, build : int);
```

Description

Initializes a new instance of the **Version** class using the specified major, minor, and build values.

Metadata restricts *major*, *minor*, and *build* to a maximum value of **System.UInt16.MaxValue** - 1. The major version number. The minor version number. The build number.

Version

Example Syntax:

ToString

[C#] public Version(int major, int minor, int build, int revision);

[C++] public: Version(int major, int minor, int build, int revision);

[VB] Public Sub New(ByVal major As Integer, ByVal minor As Integer, ByVal
build As Integer, ByVal revision As Integer)

[JScript] public function Version(major : int, minor : int, build : int, revision : int);

Initializes a new instance of the **Version** class with the specified major, minor,
build, and revision numbers.

Description

Initializes a new instance of the **Version** class with the specified major,
minor, build, and revision numbers.

Metadata restricts *major* , *minor* , *build* , and *revision* to a maximum of
System.UInt16.MaxValue - 1. The major version number. The minor version
number. The build number. The revision number.

Build

ToString

[C#] public int Build {get;}

[C++] public: __property int get_Build();

[VB] Public ReadOnly Property Build As Integer

[JScript] public function get Build() : int;

Description

Gets the value of the build component of the version number for this instance.

For example, if the version number is 6.2.1.3, the build number is 1. If the version number is 6.2, the build number is undefined.

Major

ToString

[C#] public int Major {get;}

[C++] public: __property int get_Major();

[VB] Public ReadOnly Property Major As Integer

[JScript] public function get Major() : int;

Description

Gets the value of the major component of the version number for this instance.

For example, if the version number is 6.2, the major version is 6.

Minor

ToString

[C#] public int Minor {get;}

[C++] public: __property int get_Minor();

[VB] Public ReadOnly Property Minor As Integer

[JScript] public function get Minor() : int;

Description

Gets the value of the minor component of the version number for this instance.

For example, if the version number is 6.2, the minor version is 2.

Revision

ToString

[C#] public int Revision {get;}

[C++] public: __property int get_Revision();

[VB] Public ReadOnly Property Revision As Integer

[JScript] public function get Revision() : int;

Description

Gets the value of the revision component of the version number for this instance.

For example, if the version number is 6.2.1.3, the revision number is 3. If the version number is 6.2, the revision number is undefined.

Clone

[C#] public object Clone();

[C++] public: __sealed Object* Clone();

[VB] NotOverridable Public Function Clone() As Object

[JScript] public function Clone() : Object;

Description

1 Returns a new **Version** object whose value is the same as this instance.

2 *Return Value:* A copy of this **Version** object.

3 CompareTo

4
5 [C#] public int CompareTo(object version);

6 [C++] public: __sealed int CompareTo(Object* version);

7 [VB] NotOverridable Public Function CompareTo(ByVal version As Object) As

8 Integer

9 [JScript] public function CompareTo(version : Object) : int;

10
11 *Description*

12 Compares this instance to a specified object and returns an indication of
13 their relative values.

14 *Return Value:* Return Value Description Less than zero This instance is prior to
15 *version* .

16 The components of **Version** in decreasing order of importance are: major,
17 minor, build, and revision. An unknown component is assumed to be older than
18 any known component. An object to compare, or **null**.

19 Equals

20
21 [C#] public override bool Equals(object obj);

22 [C++] public: bool Equals(Object* obj);

23 [VB] Overrides Public Function Equals(ByVal obj As Object) As Boolean

24 [JScript] public override function Equals(obj : Object) : Boolean;

Description

Returns a value indicating whether this instance is equal to a specified object.

Return Value: **true** if this instance and *version* are both **Version** objects, and every component of this instance matches the corresponding component of *version* ; otherwise, **false** . An object to compare with this instance.

GetHashCode

[C#] public override int GetHashCode();

[C++] public: int GetHashCode();

[VB] Overrides Public Function GetHashCode() As Integer

[JScript] public override function GetHashCode() : int;

Description

Returns a hash code for this instance.

Return Value: A 32-bit signed integer hash code.

op_Equality

[C#] public static bool operator ==(Version v1, Version v2);

[C++] public: static bool op_Equality(Version* v1, Version* v2);

[VB] returnValue = Version.op_Equality(v1, v2)

[JScript] returnValue = v1 == v2;

Description

Determines whether two specified instances of **Version** are equal.

Return Value: **true** if *v1* equals *v2* ; otherwise **false** . The first instance of **Version**.

The second instance of **Version**.

op_GreaterThan

[C#] public static bool operator >(Version v1, Version v2);

[C++] public: static bool op_GreaterThan(Version* v1, Version* v2);

[VB] returnValue = Version.op_GreaterThan(v1, v2)

[JScript] returnValue = v1 > v2;

Description

Determines whether the first specified instance of **Version** is greater than the second specified instance of **Version** .

Return Value: **true** if *v1* is greater than *v2* ; otherwise **false** . The first instance of **Version**. The second instance of **Version**.

op_GreaterThanOrEqual

[C#] public static bool operator >=(Version v1, Version v2);

[C++] public: static bool op_GreaterThanOrEqual(Version* v1, Version* v2);

[VB] returnValue = Version.op_GreaterThanOrEqual(v1, v2)

[JScript] returnValue = v1 >= v2;

Description

Determines whether the first specified instance of **Version** is greater than or equal to the second instance of **Version** .

1 *Return Value:* **true** if *v1* is greater than or equal to *v2* ; otherwise **false** . The first
2 instance of **Version**. The second instance of **Version**.

3 `op_Inequality`

4
5 [C#] public static bool operator !=(Version v1, Version v2);

6 [C++] public: static bool op_Inequality(Version* v1, Version* v2);

7 [VB] returnValue = Version.op_Inequality(v1, v2)

8 [JScript] returnValue = v1 != v2;

9
10 *Description*

11 Determines whether two specified instances of **Version** are not equal.

12 *Return Value:* **true** if *v1* does not equal *v2* ; otherwise **false** . The first instance of
13 **Version**. The second instance of **Version**.

14 `op_LessThan`

15
16 [C#] public static bool operator

17 [C++] public: static bool op_LessThan(Version* v1, Version* v2);

18 [VB] returnValue = Version.op_LessThan(v1, v2)

19 [JScript] returnValue = v1 < v2;

20
21 *Description*

22 Determines whether the first specified instance of **Version** is less than the
23 second specified instance of **Version** .

24 *Return Value:* **true** if *v1* is less than *v2* ; otherwise **false** . The first instance of
25 **Version**. The second instance of **Version**.

op_LessThanOrEqual

[C#] public static bool operator <=(Version v1, Version v2);

[C++] public: static bool op_LessThanOrEqual(Version* v1, Version* v2);

[VB] returnValue = Version.op_LessThanOrEqual(v1, v2)

[JScript] returnValue = v1 <= v2;

Description

Determines whether the first specified instance of **Version** is less than or equal to the second instance of **Version** .

Return Value: **true** if *v1* is less than or equal to *v2* ; otherwise **false** . The first instance of **Version**. The second instance of **Version**.

ToString

[C#] public override string ToString();

[C++] public: String* ToString();

[VB] Overrides Public Function ToString() As String

[JScript] public override function ToString() : String; Converts the value of this instance to its equivalent **String** representation.

Description

Converts the value of this instance to its equivalent **String** representation.

Return Value: The **System.String** representation of the values of the major, minor, build, and revision components of this instance, as depicted in the following format. Each component is separated by a period character ('.'). Square brackets '['

and '])' indicate a component that will not appear in the return value if the component is not defined: major.minor[.build[.revision]] For example, if you create an instance of **Version** using the constructor **Version(1,1)** , the returned string is "1.1". If you create an instance of **Version** using the constructor **Version(1,3,4,2)** , the returned string is "1.3.4.2".

ToString

[C#] public string ToString(int fieldCount);

[C++] public: String* ToString(int fieldCount);

[VB] Public Function ToString(ByVal fieldCount As Integer) As String

[JScript] public function ToString(fieldCount : int) : String;

Description

Converts the value of this instance to its equivalent **String** representation.

A specified count indicates the number of components to return.

Return Value: The **System.String** representation of the values of the major, minor, build, and revision components of this instance, each separated by a period character ('.'). The number of components to return.

Void structure (System)

ToString

Description

Indicates a method that does not return a value; that is, the method has the void return type.

This class is used in the **System.Reflection** namespace. This class has no members, and you cannot create an instance of this class.

WeakReference class (System)

ToString

Description

Represents a "weak reference", which references an object while still allowing it to be garbage collected.

The common language runtime "garbage collection" mechanism reclaims inaccessible (that is, "unreachable") memory allocated to an object. An object becomes unreachable if all references to it become invalid, for example, by setting those references to **null**.

WeakReference

Example Syntax:

ToString

[C#] public WeakReference(object target);

[C++] public: WeakReference(Object* target);

[VB] Public Sub New(ByVal target As Object)

[JScript] public function WeakReference(target : Object); Initializes a new instance of the **WeakReference** class.

Description

1 Initializes a new instance of the **WeakReference** class, referencing the
2 specified object.

3 This constructor creates a short weak reference to *target* . The object to
4 track.

5 WeakReference

6 *Example Syntax:*

7 ToString

8
9 [C#] public WeakReference(object target, bool trackResurrection);

10 [C++] public: WeakReference(Object* target, bool trackResurrection);

11 [VB] Public Sub New(ByVal target As Object, ByVal trackResurrection As
12 Boolean)

13 [JScript] public function WeakReference(target : Object, trackResurrection :
14 Boolean);

15
16 *Description*

17 Initializes a new instance of the **WeakReference** class, referencing the
18 specified object and using the specified resurrection tracking.

19 If *trackResurrection* is **false** , a short weak reference is created. If
20 *trackResurrection* is **true** , a long weak reference is created. Typically, only short
21 weak references are needed. An object to track. Indicates when to stop tracking the
22 object. If **true** , the object is tracked after finalization; if **false** , the object is only
23 tracked until finalization.

24 WeakReference

25 *Example Syntax:*

ToString

[C#] protected WeakReference(SerializationInfo info, StreamingContext context);

[C++] protected: WeakReference(SerializationInfo* info, StreamingContext context);

[VB] Protected Sub New(ByVal info As SerializationInfo, ByVal context As StreamingContext)

[JScript] protected function WeakReference(info : SerializationInfo, context : StreamingContext);

Description

Initializes a new instance of the **System.WeakReference** class, using the specified serialization and stream information.

The *context* parameter is reserved, and does not currently participate in this operation. An object that holds all the data needed to serialize or deserialize this instance. (Reserved) Describes the source and destination of the serialized stream specified by *info*.

IsAlive

ToString

[C#] public virtual bool IsAlive {get;}

[C++] public: __property virtual bool get_IsAlive();

[VB] Overridable Public ReadOnly Property IsAlive As Boolean

[JScript] public function get IsAlive() : Boolean;

1
2 *Description*

3 Gets an indication whether the object referenced by this instance has been
4 garbage collected.

5 **WeakReferenceException** is thrown if the object referenced by this
6 instance is invalid. This can occur if there is an attempt to resurrect the referenced
7 object after it has been finalized.

8 Target

9 ToString

10
11 [C#] public virtual object Target {get; set;}

12 [C++] public: __property virtual Object* get_Target();public: __property virtual
13 void set_Target(Object*);

14 [VB] Overridable Public Property Target As Object

15 [JScript] public function get Target() : Object;public function set Target(Object);

16
17 *Description*

18 Gets or sets the object (the "target") referenced by this instance.

19 **WeakReferenceException** will be thrown if the reference to the object
20 referenced by this instance is invalid. This can occur if there is an attempt to
21 resurrect the referenced object after it has been finalized.

22 TrackResurrection

23 ToString

24
25 [C#] public virtual bool TrackResurrection {get;}

1 [C++] public: __property virtual bool get_TrackResurrection();

2 [VB] Overridable Public ReadOnly Property TrackResurrection As Boolean

3 [JScript] public function get TrackResurrection() : Boolean;

4
5 *Description*

6 Gets an indication whether the object referenced by this instance is tracked
7 after it is garbage collected and finalized.

8 Finalize

9
10 [C#] ~WeakReference();

11 [C++] ~WeakReference();

12 [VB] Overrides Protected Sub Finalize()

13 [JScript] protected override function Finalize();

14
15 *Description*

16 Frees any resources allocated by this instance.

17 GetObjectData

18
19 [C#] public virtual void GetObjectData(SerializationInfo info, StreamingContext
20 context);

21 [C++] public: virtual void GetObjectData(SerializationInfo* info,
22 StreamingContext context);

23 [VB] Overridable Public Sub GetObjectData(ByVal info As SerializationInfo,
24 ByVal context As StreamingContext)

25 [JScript] public function GetObjectData(info : SerializationInfo, context :

1 StreamingContext);

2
3 *Description*

4 Populates a **SerializationInfo** object with all the data needed to serialize
5 this instance.

6 This method stores all the information in *info* necessary to serialize this
7 instance. A **SerializationInfo** object. (Reserved) The location where serialized
8 data will be stored and retrieved.

9 Uri class (System)

10 ToString

11
12
13 *Description*

14 Provides an object representation of a uniform resource identifier (URI)
15 and easy access to the parts of the URI.

16 A URI is a compact representation of a resource available to your
17 application on the Internet. The **System.Uri** class defines the properties and
18 methods for handling URIs, including parsing, comparing, and combining. The
19 **System.Uri** class properties are read-only, to modify a **System.Uri** instance use
20 the **System.UriBuilder** class.

21 ToString

22
23 [C#] public static readonly string SchemeDelimiter;

24 [C++] public: static String* SchemeDelimiter;

25 [VB] Public Shared ReadOnly SchemeDelimiter As String

1 [JScript] public static var SchemeDelimiter : String;

3 *Description*

4 Specifies the characters that separate the communication protocol scheme
5 from the address portion of the URI. This field is read-only.

6 ToString

8 [C#] public static readonly string UriSchemeFile;

9 [C++] public: static String* UriSchemeFile;

10 [VB] Public Shared ReadOnly UriSchemeFile As String

11 [JScript] public static var UriSchemeFile : String;

13 *Description*

14 Specifies that the URI is a pointer to a file. This field is read-only.

15 ToString

17 [C#] public static readonly string UriSchemeFtp;

18 [C++] public: static String* UriSchemeFtp;

19 [VB] Public Shared ReadOnly UriSchemeFtp As String

20 [JScript] public static var UriSchemeFtp : String;

22 *Description*

23 Specifies that the URI is accessed through the File Transfer Protocol (FTP).

24 This field is read-only.

25 ToString

1
2 [C#] public static readonly string UriSchemeGopher;

3 [C++] public: static String* UriSchemeGopher;

4 [VB] Public Shared ReadOnly UriSchemeGopher As String

5 [JScript] public static var UriSchemeGopher : String;

6
7 *Description*

8 Specifies that the URI is accessed through the Gopher protocol. This field
9 is read-only.

10 ToString

11
12 [C#] public static readonly string UriSchemeHttp;

13 [C++] public: static String* UriSchemeHttp;

14 [VB] Public Shared ReadOnly UriSchemeHttp As String

15 [JScript] public static var UriSchemeHttp : String;

16
17 *Description*

18 Specifies that the URI is accessed through the Hypertext Transfer Protocol
19 (HTTP). This field is read-only.

20 ToString

21
22 [C#] public static readonly string UriSchemeHttps;

23 [C++] public: static String* UriSchemeHttps;

24 [VB] Public Shared ReadOnly UriSchemeHttps As String

25 [JScript] public static var UriSchemeHttps : String;

1
2 *Description*

3 Specifies that the URI is accessed through the Secure Hypertext Transfer
4 Protocol (HTTPS). This field is read-only.

5 ToString

6
7 [C#] public static readonly string UriSchemeMailto;

8 [C++] public: static String* UriSchemeMailto;

9 [VB] Public Shared ReadOnly UriSchemeMailto As String

10 [JScript] public static var UriSchemeMailto : String;

11
12 *Description*

13 Specifies that the URI is an email address and is accessed through the
14 Simple Network Mail Protocol (SNMP). This field is read-only.

15 ToString

16
17 [C#] public static readonly string UriSchemeNews;

18 [C++] public: static String* UriSchemeNews;

19 [VB] Public Shared ReadOnly UriSchemeNews As String

20 [JScript] public static var UriSchemeNews : String;

21
22 *Description*

23 Specifies that the URI is an Internet news group and is accessed through the
24 Network News Transport Protocol (NNTP). This field is read-only.

25 ToString

1
2 [C#] public static readonly string UriSchemeNntp;

3 [C++] public: static String* UriSchemeNntp;

4 [VB] Public Shared ReadOnly UriSchemeNntp As String

5 [JScript] public static var UriSchemeNntp : String;

6
7 *Description*

8 Specifies that the URI is an Internet news group and is accessed through the
9 Network News Transport Protocol (NNTP). This field is read-only.

10 Uri

11 *Example Syntax:*

12 ToString

13
14 [C#] public Uri(string uriString);

15 [C++] public: Uri(String* uriString);

16 [VB] Public Sub New(ByVal uriString As String)

17 [JScript] public function Uri(uriString : String); Initializes a new instance of the
18 **System.Uri** class.

19
20 *Description*

21 Initializes a new instance of the **System.Uri** class with the specified URI.

22 This constructor creates a **System.Uri** instance from a URI string. It parses
23 the URI, puts it in canonical format, and makes any required escape encodings. A
24 URI.

25 Uri

Example Syntax:

ToString

[C#] protected Uri(SerializationInfo serializationInfo, StreamingContext
streamingContext);

[C++] protected: Uri(SerializationInfo* serializationInfo, StreamingContext
streamingContext);

[VB] Protected Sub New(ByVal serializationInfo As SerializationInfo, ByVal
streamingContext As StreamingContext)

[JScript] protected function Uri(serializationInfo : SerializationInfo,
streamingContext : StreamingContext);

Description

Initializes a new instance of the **System.Uri** class from the specified
instances of the **System.Runtime.Serialization.SerializationInfo** and
System.Runtime.Serialization.StreamingContext classes.

This constructor implements the
System.Runtime.Serialization.ISerializable interface for the **System.Uri** class.
An instance of the **System.Runtime.Serialization.SerializationInfo** class
containing the information required to serialize the new **System.Uri** instance. An
instance of the **System.Runtime.Serialization.StreamingContext** class
containing the source of the serialized stream associated with the new **System.Uri**
instance.

Uri

Example Syntax:

ToString

```
[C#] public Uri(string uriString, bool dontEscape);  
[C++] public: Uri(String* uriString, bool dontEscape);  
[VB] Public Sub New(ByVal uriString As String, ByVal dontEscape As Boolean)  
[JScript] public function Uri(uriString : String, dontEscape : Boolean);
```

Description

Initializes a new instance of the **System.Uri** class with the specified URI, with control of character escaping.

This constructor creates a **System.Uri** instance from a URI string. It parses the URI, and puts it in canonical format. The caller must set *dontEscape* to **true** if the URI is already escaped, or false if the constructor should transform the URI to its escaped encoding. The URI. **true** if the URI contains escape characters; otherwise, **false**.

Uri

Example Syntax:

ToString

```
[C#] public Uri(Uri baseUri, string relativeUri);  
[C++] public: Uri(Uri* baseUri, String* relativeUri);  
[VB] Public Sub New(ByVal baseUri As Uri, ByVal relativeUri As String)  
[JScript] public function Uri(baseUri : Uri, relativeUri : String);
```

Description

1 Initializes a new instance of the **System.Uri** class based on the specified
2 base and relative URIs.

3 This constructor creates a **System.Uri** instance by combining the *baseUri*
4 and the *relativeUri* . If *relativeUri* is an absolute URI (containing a scheme,
5 hostname, and optionally a port number) the **System.Uri** instance is created using
6 only *relativeUri* . The base URI. The relative URI to add to the base URI.

7 Uri

8 *Example Syntax:*

9 ToString

10
11 [C#] public Uri(Uri baseUri, string relativeUri, bool dontEscape);

12 [C++] public: Uri(Uri* baseUri, String* relativeUri, bool dontEscape);

13 [VB] Public Sub New(ByVal baseUri As Uri, ByVal relativeUri As String, ByVal
14 dontEscape As Boolean)

15 [JScript] public function Uri(baseUri : Uri, relativeUri : String, dontEscape :
16 Boolean);

17
18 *Description*

19 Initializes a new instance of the **System.Uri** class based on the specified
20 base and relative URIs, with control of character escaping.

21 This constructor creates a **System.Uri** instance by combining *baseUri* and
22 *relativeUri* . If the URI passed in *relativeUri* is an absolute URI (containing a
23 scheme, hostname, and optionally a port number) the **System.Uri** instance is
24 created using only *relativeUri* . The caller must set *dontEscape* to **true** if the URI
25

is already escaped. The base URI. The relative URI to add to the base URI. **true** if the URI contains escape characters; otherwise, **false**.

AbsolutePath

ToString

```
[C#] public string AbsolutePath {get;}
```

```
[C++] public: __property String* get_AbsolutePath();
```

```
[VB] Public ReadOnly Property AbsolutePath As String
```

```
[JScript] public function get AbsolutePath() : String;
```

Description

Gets the absolute path of the URI.

The **System.Uri.AbsolutePath** property contains the path information that the server uses to resolve requests for information. Typically this is the path to the desired information on the server's file system, although it also can indicate the application or script the server must run to provide the information.

AbsoluteUri

ToString

```
[C#] public string AbsoluteUri {get;}
```

```
[C++] public: __property String* get_AbsoluteUri();
```

```
[VB] Public ReadOnly Property AbsoluteUri As String
```

```
[JScript] public function get AbsoluteUri() : String;
```

Description

Gets the absolute URI.

The **System.Uri.AbsoluteUri** property includes the entire URI stored in the **System.Uri** instance, including all fragments and query strings.

Authority

ToString

[C#] public string Authority {get;}

[C++] public: __property String* get_Authority();

[VB] Public ReadOnly Property Authority As String

[JScript] public function get Authority() : String;

Description

Gets the Domain Name System (DNS) host name or IP address and the port number for a server.

The Authority property returns the fully qualified DNS host name or IP address of the server specified in the URI, along with the port number for non-default ports.

Fragment

ToString

[C#] public string Fragment {get;}

[C++] public: __property String* get_Fragment();

[VB] Public ReadOnly Property Fragment As String

[JScript] public function get Fragment() : String;

Description

Gets the escaped fragment.

The **System.Uri.Fragment** property gets any text following a fragment marker (#) in the URI, including the fragment marker itself. Given the URI `http://www.contoso.com/index.htm#main`, the fragment property would get `#main`.

Host

ToString

```
[C#] public string Host {get;}
```

```
[C++] public: __property String* get_Host();
```

```
[VB] Public ReadOnly Property Host As String
```

```
[JScript] public function get Host() : String;
```

Description

Gets the Domain Name System (DNS) host name, or IP address of the server specified in the URI.

The **System.Uri.Host** property gets the fully qualified DNS host name or IP address of the server specified in the URI.

HostNameType

ToString

```
[C#] public UriHostNameType HostNameType {get;}
```

```
[C++] public: __property UriHostNameType get_HostNameType();
```

1 [VB] Public ReadOnly Property HostNameType As UriHostNameType

2 [JScript] public function get HostNameType() : UriHostNameType;

3
4 *Description*

5 Returns the type of the host name specified in the URI.

6 IsDefaultPort

7 ToString

8
9 [C#] public bool IsDefaultPort {get;}

10 [C++] public: __property bool get_IsDefaultPort();

11 [VB] Public ReadOnly Property IsDefaultPort As Boolean

12 [JScript] public function get IsDefaultPort() : Boolean;

13
14 *Description*

15 Gets a value indicating whether the port value of the URI is the default for
16 this scheme.

17 IsFile

18 ToString

19
20 [C#] public bool IsFile {get;}

21 [C++] public: __property bool get_IsFile();

22 [VB] Public ReadOnly Property IsFile As Boolean

23 [JScript] public function get IsFile() : Boolean;

24
25 *Description*

Gets a value indicating whether the specified **System.Uri** is a file URI.

The **System.Uri.IsFile** property is **true** when the **System.Uri.Scheme** property equals **System.Uri.UriSchemeFile** .

IsLoopback

ToString

[C#] public bool IsLoopback {get;}

[C++] public: __property bool get_IsLoopback();

[VB] Public ReadOnly Property IsLoopback As Boolean

[JScript] public function get IsLoopback() : Boolean;

Description

Gets a value indicating whether the specified **System.Uri** references the local host.

IsUnc

ToString

[C#] public bool IsUnc {get;}

[C++] public: __property bool get_IsUnc();

[VB] Public ReadOnly Property IsUnc As Boolean

[JScript] public function get IsUnc() : Boolean;

Description

Gets a value indicating whether the specified **System.Uri** is a universal naming convention (UNC) path.

The **System.Uri.IsUnc** property is **true** if the specified **System.Uri** instance is a UNC path (suc as \\server\folder).

LocalPath

ToString

[C#] public string LocalPath {get;}

[C++] public: __property String* get_LocalPath();

[VB] Public ReadOnly Property LocalPath As String

[JScript] public function get LocalPath() : String;

Description

Gets a local operating-system representation of a file name.

PathAndQuery

ToString

[C#] public string PathAndQuery {get;}

[C++] public: __property String* get_PathAndQuery();

[VB] Public ReadOnly Property PathAndQuery As String

[JScript] public function get PathAndQuery() : String;

Description

Gets the **System.Uri.AbsolutePath** and **System.Uri.Query** properties separated by a question mark (?).

The **System.Uri.PathAndQuery** property contains the absolute path on the server and the query information sent with the request. It is identical to concatenating the **System.Uri.AbsolutePath** and **System.Uri.Query** properties.

Port

ToString

[C#] public int Port {get;}

[C++] public: __property int get_Port();

[VB] Public ReadOnly Property Port As Integer

[JScript] public function get Port() : int;

Description

Gets the port number of the specified URI.

The port number defines the protocol port used for contacting the server referenced in the URI. If a port is not specified as part of the URI, the **System.Uri.Port** property returns the default value for the protocol.

Query

ToString

[C#] public string Query {get;}

[C++] public: __property String* get_Query();

[VB] Public ReadOnly Property Query As String

[JScript] public function get Query() : String;

Description

Gets any query information included in the specified URI.

The **System.Uri.Query** property contains any query information included in the URI. Query information is separated from the path information by a question mark (?) and continues to the end of the URI. The query information returned includes the leading question mark.

Scheme

ToString

[C#] public string Scheme {get;}

[C++] public: __property String* get_Scheme();

[VB] Public ReadOnly Property Scheme As String

[JScript] public function get Scheme() : String;

Description

Gets the scheme name of the specified URI.

The following table lists the valid scheme names for the **System.Uri.Scheme** property.

Segments

ToString

[C#] public string[] Segments {get;}

[C++] public: __property String* get_Segments();

[VB] Public ReadOnly Property Segments As String ()

[JScript] public function get Segments() : String[];

1
2 *Description*

3 Gets an array of the segments that make up the specified URI.

4 UserEscaped

5 ToString

6
7 [C#] public bool UserEscaped {get;}

8 [C++] public: __property bool get_UserEscaped();

9 [VB] Public ReadOnly Property UserEscaped As Boolean

10 [JScript] public function get UserEscaped() : Boolean;

11
12 *Description*

13 Indicates that the URI string was escaped before the **System.Uri** instance
14 was created.

15 The **System.Uri.UserEscaped** property is set to **true** to indicate that the
16 string used to create the **System.Uri** instance was escaped before it was passed to
17 the constructor; that is, the *dontEscape* parameter of the constructor call was set to
18 **true** .

19 UserInfo

20 ToString

21
22 [C#] public string UserInfo {get;}

23 [C++] public: __property String* get_UserInfo();

24 [VB] Public ReadOnly Property UserInfo As String

25 [JScript] public function get UserInfo() : String;

1
2 *Description*

3 Gets the user name, password, and other user-specific information
4 associated with the specified URI.

5 Canonicalize

6
7 [C#] protected virtual void Canonicalize();
8 [C++] protected: virtual void Canonicalize();
9 [VB] Overridable Protected Sub Canonicalize()
10 [JScript] protected function Canonicalize();
11

12 *Description*

13 CheckHostName

14
15 [C#] public static UriHostNameType CheckHostName(string name);
16 [C++] public: static UriHostNameType CheckHostName(String* name);
17 [VB] Public Shared Function CheckHostName(ByVal name As String) As
18 UriHostNameType
19 [JScript] public static function CheckHostName(name : String) :
20 UriHostNameType;
21

22 *Description*

23 Determines whether the specified host name is valid.

24 *Return Value:* **true** if the host name is valid; otherwise, **false** .
25

1 The **System.Uri.CheckHostName(System.String)** method checks that the
2 host name provided meets the requirements for a valid Internet host name. It does
3 not, however, perform a host-name lookup to verify the existence of the host. The
4 host name to validate.

5 CheckSchemeName

6
7 [C#] public static bool CheckSchemeName(string schemeName);
8 [C++] public: static bool CheckSchemeName(String* schemeName);
9 [VB] Public Shared Function CheckSchemeName(ByVal schemeName As String)
10 As Boolean
11 [JavaScript] public static function CheckSchemeName(schemeName : String) :
12 Boolean;

13 Description

14 Determines whether the specified scheme name is valid.

15 Return Value: **true** if the scheme name is valid; otherwise, **false** .

16 The CheckSchemeName method checks the scheme name for validity
17 according to RFC 2396. The scheme name must begin with a letter, and must
18 contain only letters, digits, and the characters ".", "+", or "-". The scheme name to
19 validate.
20

21 CheckSecurity

22
23 [C#] protected virtual void CheckSecurity();
24 [C++] protected: virtual void CheckSecurity();
25 [VB] Overridable Protected Sub CheckSecurity()

1 [JScript] protected function CheckSecurity();

3 *Description*

4 Equals

6 [C#] public override bool Equals(object comparand);

7 [C++] public: bool Equals(Object* comparand);

8 [VB] Overrides Public Function Equals(ByVal comparand As Object) As Boolean

9 [JScript] public override function Equals(comparand : Object) : Boolean;

11 *Description*

12 Compares two **System.Uri** instances for equality.

13 *Return Value:* **true** if the two **System.Uri** instances contain the same URI;
14 otherwise, **false** .

15 The **System.Uri.Equals(System.Object)** method compares two URI
16 instances without regard to any fragments that they might contain. For instance,
17 given the URIs <http://www.contoso.com/index.htm#search> and
18 <http://www.contoso.com/index.htm> the **System.Uri.Equals(System.Object)**
19 method would return **true** . The **System.Uri** instance to compare with the current
20 instance.

21 Escape

23 [C#] protected virtual void Escape();

24 [C++] protected: virtual void Escape();

25 [VB] Overridable Protected Sub Escape()

1 [JScript] protected function Escape();

3 *Description*

4 EscapeString

6 [C#] protected static string EscapeString(string str);

7 [C++] protected: static String* EscapeString(String* str);

8 [VB] Protected Shared Function EscapeString(ByVal str As String) As String

9 [JScript] protected static function EscapeString(str : String) : String;

11 *Description*

12 Converts a string to its escaped representation.

13 *Return Value:* The escaped representation of the string.

14 The **System.Uri.EscapeString(System.String)** method converts all
15 characters with an ASCII value greater than 127 to hexadecimal representation.

16 The string to transform to its escaped representation.

17 FromHex

19 [C#] public static int FromHex(char digit);

20 [C++] public: static int FromHex(__wchar_t digit);

21 [VB] Public Shared Function FromHex(ByVal digit As Char) As Integer

22 [JScript] public static function FromHex(digit : Char) : int;

24 *Description*

1 Returns the decimal value of a hexadecimal digit.

2 *Return Value:* A number from 1 - 15 that corresponds to the specified hexadecimal
3 digit.

4 The **System.Uri.FromHex(System.Char)** method converts a character
5 representing a hexadecimal digit (0-9, a-f, A-F) to its decimal value (0-15). If digit
6 is not a valid hexadecimal digit, **System.ArgumentException** is thrown. The
7 hexadecimal digit (0-9, a-f, A-F) to convert.

8 GetHashCode

9
10 [C#] public override int GetHashCode();

11 [C++] public: int GetHashCode();

12 [VB] Overrides Public Function GetHashCode() As Integer

13 [JScript] public override function GetHashCode() : int;

14 15 *Description*

16 Returns the hash code for the specified URI.

17 *Return Value:* The hash value generated for the URI.

18 The **System.Uri.GetHashCode** method generates the URI's hash value
19 without including any fragment identifiers. For example, the URIs
20 <http://www.contoso.com/index.htm#search> and
21 <http://www.contoso.com/index.htm> yield the same hash code.

22 GetLeftPart

23
24 [C#] public string GetLeftPart(UriPartial part);

25 [C++] public: String* GetLeftPart(UriPartial part);

1 [VB] Public Function GetLeftPart(ByVal part As UriPartial) As String

2 [JScript] public function GetLeftPart(part : UriPartial) : String;

3
4 *Description*

5 Returns the specified portion of a URI.

6 *Return Value:* A string containing the specified portion of the URI.

7 The **System.Uri.GetLeftPart(System.UriPartial)** method returns a string
8 containing the left-most portion of the URI, ending with the portion specified by
9 *part* . The string returned includes delimiters but does not include any fragments
10 or queries or their delimiters, except in certain cases. One of the
11 **System.UriPartial** values that specifies the the end of the portion of the URI to
12 return.

13 HexEscape

14
15 [C#] public static string HexEscape(char character);

16 [C++] public: static String* HexEscape(__wchar_t character);

17 [VB] Public Shared Function HexEscape(ByVal character As Char) As String

18 [JScript] public static function HexEscape(character : Char) : String;

19
20 *Description*

21 Converts a specified character into its hexadecimal equivalent.

22 *Return Value:* The hexadecimal representation of the specified character. The
23 character to convert to hexadecimal representation.

24 HexUnescape

```

1
2 [C#] public static char HexUnescape(string pattern, ref int index);
3 [C++] public: static __wchar_t HexUnescape(String* pattern, int* index);
4 [VB] Public Shared Function HexUnescape(ByVal pattern As String, ByRef index
5 As Integer) As Char
6 [JScript] public static function HexUnescape(pattern : String, index : int) : Char;
7

```

Description

Converts a specified hexadecimal representation of a character to the character.

Return Value: The character represented by the hexadecimal encoding at position *index* . If the character at *index* is not hexadecimal encoded, the character at *index* is returned. The value of *index* is incremented to point to the character following the one returned. The hexadecimal representation of a character. The location in *pattern* where the hexadecimal representation of a character begins.

IsBadFileSystemCharacter

```

18 [C#] protected virtual bool IsBadFileSystemCharacter(char character);
19 [C++] protected: virtual bool IsBadFileSystemCharacter(__wchar_t character);
20 [VB] Overridable Protected Function IsBadFileSystemCharacter(ByVal character
21 As Char) As Boolean
22 [JScript] protected function IsBadFileSystemCharacter(character : Char) :
23 Boolean;
24

```

Description

IsExcludedCharacter

[C#] protected static bool IsExcludedCharacter(char character);
[C++] protected: static bool IsExcludedCharacter(__wchar_t character);
[VB] Protected Shared Function IsExcludedCharacter(ByVal character As Char)
As Boolean
[JScript] protected static function IsExcludedCharacter(character : Char) :
Boolean;

Description

IsHexDigit

[C#] public static bool IsHexDigit(char character);
[C++] public: static bool IsHexDigit(__wchar_t character);
[VB] Public Shared Function IsHexDigit(ByVal character As Char) As Boolean
[JScript] public static function IsHexDigit(character : Char) : Boolean;

Description

Determines whether a specified character is a valid hexadecimal digit.

Return Value: **true** if the character is a valid hexadecimal digit; otherwise **false** .

Hexadecimal digits are the digits 0-9 and the letters A-F or a-f. The character to validate.

IsHexEncoding

[C#] public static bool IsHexEncoding(string pattern, int index);

[C++] public: static bool IsHexEncoding(String* pattern, int index);

[VB] Public Shared Function IsHexEncoding(ByVal pattern As String, ByVal index As Integer) As Boolean

[JScript] public static function IsHexEncoding(pattern : String, index : int) : Boolean;

Description

Determines whether a string is hex encoded.

Return Value: **true** if *pattern* is hex encoded at the specified location; otherwise, **false**.

The **System.Uri.IsHexEncoding(System.String, System.Int32)** method checks for hex encoding that follows the pattern "%hexhex" in a string, where "hex" is a digit from 0-9 or or a letter from A-F (case-insensitive). The string to check. The location in *pattern* to check for hex encoding.

IsReservedCharacter

[C#] protected virtual bool IsReservedCharacter(char character);

[C++] protected: virtual bool IsReservedCharacter(__wchar_t character);

[VB] Overridable Protected Function IsReservedCharacter(ByVal character As Char) As Boolean

[JScript] protected function IsReservedCharacter(character : Char) : Boolean;

Description

MakeRelative

1
2 [C#] public string MakeRelative(Uri toUri);

3 [C++] public: String* MakeRelative(Uri* toUri);

4 [VB] Public Function MakeRelative(ByVal toUri As Uri) As String

5 [JScript] public function MakeRelative(toUri : Uri) : String;

6
7 *Description*

8 Determines the difference between two **System.Uri** instances.

9 *Return Value:* If the two URIs are the same except for the path information, then
10 that difference; if the two have additional differences, the absolute URI of *toUri* .
11 The URI to compare to the current URI.

12 **Parse**

13
14 [C#] protected virtual void Parse();

15 [C++] protected: virtual void Parse();

16 [VB] Overridable Protected Sub Parse()

17 [JScript] protected function Parse();

18
19 *Description*

20 **ISerializable.GetObjectData**

21
22 [C#] void ISerializable.GetObjectData(SerializationInfo serializationInfo,
23 StreamingContext streamingContext);

24 [C++] void ISerializable::GetObjectData(SerializationInfo* serializationInfo,
25 StreamingContext streamingContext);

[VB] Sub GetObjectData(ByVal serializationInfo As SerializationInfo, ByVal
streamingContext As StreamingContext) Implements ISerializable.GetObjectData
[JScript] function ISerializable.GetObjectData(serializationInfo : SerializationInfo,
streamingContext : StreamingContext);

ToString

[C#] public override string ToString();

[C++] public: String* ToString();

[VB] Overrides Public Function ToString() As String

[JScript] public override function ToString() : String;

Description

Returns the display string for the specified **System.Uri** instance.

Return Value: The string containing the unescaped display name of the
System.Uri.

Unescape

[C#] protected virtual string Unescape(string path);

[C++] protected: virtual String* Unescape(String* path);

[VB] Overridable Protected Function Unescape(ByVal path As String) As String

[JScript] protected function Unescape(path : String) : String;

Description

UriBuilder class (System)

Unescape

1
2
3 *Description*

4 Provides a custom constructor for uniform resource indentifiers (URIs) and
5 modifies URIs for the **System.Uri** class.

6 The **System.UriBuilder** class provides a convenient way to modify the
7 contents of a **System.Uri** instance without creating a new **System.Uri** instance for
8 each modification.

9 UriBuilder

10 *Example Syntax:*

11 Unescape

12
13 [C#] public UriBuilder();

14 [C++] public: UriBuilder();

15 [VB] Public Sub New()

16 [JScript] public function UriBuilder(); Initializes a new instance of the
17 **System.UriBuilder** class.

18
19 *Description*

20 Initializes a new instance of the **System.UriBuilder** class.

21 The default constructor creates a new instance of the **System.UriBuilder**
22 class with all properties empty.

23 UriBuilder

24 *Example Syntax:*

25 Unescape


```

1  [C#] public UriBuilder(string uri);
2
3  [C++] public: UriBuilder(String* uri);
4
5  [VB] Public Sub New(ByVal uri As String)
6
7  [JScript] public function UriBuilder(uri : String);
8

```

Description

Initializes a new instance of the **System.UriBuilder** class with the specified URI.

This constructor initializes a new instance of the **System.UriBuilder** class with the **System.UriBuilder.Fragment** , **System.UriBuilder.Host** , **System.UriBuilder.Path** , **System.UriBuilder.Port** , **System.UriBuilder.Query** , **System.UriBuilder.Scheme** , and **System.UriBuilder.Uri** properties set as specified in *uri* . A URI string.

UriBuilder

Example Syntax:

Unescape

```

19 [C#] public UriBuilder(Uri uri);
20
21 [C++] public: UriBuilder(Uri* uri);
22
23 [VB] Public Sub New(ByVal uri As Uri)
24
25 [JScript] public function UriBuilder(uri : Uri);
26

```

Description

1 Initializes a new instance of the **System.UriBuilder** class with the specified
2 **System.Uri** instance.

3 This constructor initializes a new instance of the **System.UriBuilder** class
4 with the **System.UriBuilder.Fragment** , **System.UriBuilder.Host** ,
5 **System.UriBuilder.Path** , **System.UriBuilder.Port** , **System.UriBuilder.Query**
6 , **System.UriBuilder.Scheme** , and **System.UriBuilder.Uri** properties set as
7 specified in *uri* . An instance of the **System.Uri** class.

8 UriBuilder

9 *Example Syntax:*

10 Unescape

11
12 [C#] public UriBuilder(string schemeName, string hostName);

13 [C++] public: UriBuilder(String* schemeName, String* hostName);

14 [VB] Public Sub New(ByVal schemeName As String, ByVal hostName As
15 String)

16 [JScript] public function UriBuilder(schemeName : String, hostName : String);

17
18 *Description*

19 Initializes a new instance of the **System.UriBuilder** class with the specified
20 scheme and host.

21 The **System.UriBuilder** instance is initialized with the
22 **System.UriBuilder.Scheme** property set to *schemeName* and the
23 **System.UriBuilder.Host** property set to *hostName* . An Internet access protocol.
24 A DNS-style domain name or IP address.

25 UriBuilder

Example Syntax:

Unescape

```
[C#] public UriBuilder(string scheme, string host, int portNumber);  
[C++] public: UriBuilder(String* scheme, String* host, int portNumber);  
[VB] Public Sub New(ByVal scheme As String, ByVal host As String, ByVal  
portNumber As Integer)  
[JScript] public function UriBuilder(scheme : String, host : String, portNumber :  
int);
```

Description

Initializes a new instance of the **System.UriBuilder** class with the specified scheme, host, and port.

The **System.UriBuilder** instance is initialized with the **System.UriBuilder.Scheme** property set to *schemeName*, the **System.UriBuilder.Host** property set to *hostName*, and the **System.UriBuilder.Port** property set to *portNumber*. The **System.UriBuilder.Path** property is set to the slash character (/). An Internet access protocol. A DNS-style domain name or IP address. An IP port number for the service.

UriBuilder

Example Syntax:

Unescape

```
[C#] public UriBuilder(string scheme, string host, int port, string pathValue);
```

```

1 [C++] public: UriBuilder(String* scheme, String* host, int port, String*
2 pathValue);
3 [VB] Public Sub New(ByVal scheme As String, ByVal host As String, ByVal port
4 As Integer, ByVal pathValue As String)
5 [JScript] public function UriBuilder(scheme : String, host : String, port : int,
6 pathValue : String);

```

Description

Initializes a new instance of the **System.UriBuilder** class with the specified scheme, host, port number, and path.

The **System.UriBuilder** instance is initialized with the **System.UriBuilder.Scheme** property set to *schemeName*, the **System.UriBuilder.Host** property set to *hostName*, the **System.UriBuilder.Port** property set to *portNumber* and the **System.UriBuilder.Path** property set to *pathValue*. An Internet access protocol. A DNS-style domain name or IP address. An IP port number for the service. The path to the Internet resource.

UriBuilder

Example Syntax:

Unescape

```

21 [C#] public UriBuilder(string scheme, string host, int port, string path, string
22 extraValue);
23 [C++] public: UriBuilder(String* scheme, String* host, int port, String* path,
24 String* extraValue);
25 [VB] Public Sub New(ByVal scheme As String, ByVal host As String, ByVal port

```

As Integer, ByVal path As String, ByVal extraValue As String)

[JScript] public function UriBuilder(scheme : String, host : String, port : int, path : String, extraValue : String);

Description

Initializes a new instance of the **System.UriBuilder** class with the specified scheme, host, port number, path and query string or fragment identifier.

The **System.UriBuilder** instance is initialized with the **System.UriBuilder.Scheme** property set to *schemeName*, the **System.UriBuilder.Host** property set to *hostName*, the **System.UriBuilder.Port** property set to *portNumber*, and the **System.UriBuilder.Path** property is set to *pathValue*. If *extraValue* contains a number sign (#), then **System.UriBuilder.Fragment** is set to *extraValue*. If *extraValue* contains a question mark (?), then **System.UriBuilder.Query** is set to *extraValue*. An Internet access protocol. A DNS-style domain name or IP address. An IP port number for the service. The path to the Internet resource. A query string or fragment identifier.

Fragment

Unescape

[C#] public string Fragment {get; set;}

[C++] public: __property String* get_Fragment();public: __property void set_Fragment(String*);

[VB] Public Property Fragment As String

[JScript] public function get Fragment() : String;public function set

1 Fragment(String);

3 *Description*

4 Gets or sets the fragment portion of the URI.

5 The **System.UriBuilder.Fragment** property contains any text following a
6 fragment marker (#) in the URI, including the marker itself. When setting the
7 **System.UriBuilder.Fragment** property, the fragment marker is added to its value.

8 Host

9 Unescape

11 [C#] public string Host {get; set;}

12 [C++] public: __property String* get_Host();public: __property void

13 set_Host(String*);

14 [VB] Public Property Host As String

15 [JScript] public function get Host() : String;public function set Host(String);

17 *Description*

18 Gets or sets the Domain Name System (DNS) host name or IP address of a
19 server.

20 The **System.UriBuilder.Host** property contains the fully qualified DNS
21 host name or IP address (in dotted-quad notation) of the server.

22 Password

23 Unescape

25 [C#] public string Password {get; set;}

```

1  [C++] public: __property String* get_Password();public: __property void
2  set_Password(String*);
3  [VB] Public Property Password As String
4  [JScript] public function get Password() : String;public function set
5  Password(String);
6

```

Description

Gets or sets the password associated with the user accessing the URI.

Path

Unescape

```

12 [C#] public string Path {get; set;}
13 [C++] public: __property String* get_Path();public: __property void
14 set_Path(String*);
15 [VB] Public Property Path As String
16 [JScript] public function get Path() : String;public function set Path(String);
17

```

Description

Gets or sets the path to the resource referenced by the URI.

The **System.UriBuilder.Path** property contains the path information that the server uses to resolve requests for information. Typically this is the path to the desired information on the server's file system, although it also can indicate the application or script that the server must run to provide the information.

Port

Unescape

1
2 [C#] public int Port {get; set;}

3 [C++] public: __property int get_Port();public: __property void set_Port(int);

4 [VB] Public Property Port As Integer

5 [JScript] public function get Port() : int;public function set Port(int);

6
7 *Description*

8 Gets or sets the port number of the URI.

9 The port number defines the protocol port for contacting the server
10 referenced in the URI. If a port is not specified as part of the URI, the
11 **System.Uri.Port** property returns the value -1 to indicate that the server uses the
12 default value for the protocol.

13 Query

14 Unescape

15
16 [C#] public string Query {get; set;}

17 [C++] public: __property String* get_Query();public: __property void
18 set_Query(String*);

19 [VB] Public Property Query As String

20 [JScript] public function get Query() : String;public function set Query(String);

21
22 *Description*

23 Gets or sets any query information included in the URI.

24 The **System.Uri.Query** property contains any query information included
25 in the URI. Query information is separated from the path information by a

question mark (?) and continues to the end of the URI. The query information returned includes the leading question mark.

Scheme

Unescape

[C#] public string Scheme {get; set;}

[C++] public: __property String* get_Scheme();public: __property void set_Scheme(String*);

[VB] Public Property Scheme As String

[JScript] public function get Scheme() : String;public function set Scheme(String);

Description

Gets or sets the scheme name of the URI.

The following table lists the valid scheme names for the **System.Uri.Scheme** property.

Uri

Unescape

[C#] public Uri Uri {get;}

[C++] public: __property Uri* get_Uri();

[VB] Public ReadOnly Property Uri As Uri

[JScript] public function get Uri() : Uri;

Description

1 Gets the **System.Uri** instance constructed by the specified
2 **System.UriBuilder** instance.

3 The **System.UriBuilder.Uri** property contains the **System.Uri** created by
4 the **System.UriBuilder** . Any changes made to the **System.UriBuilder** properties
5 are reflected in the **System.UriBuilder.Uri** property.

6 UserName

7 Unescape

8
9 [C#] public string UserName {get; set;}

10 [C++] public: __property String* get_UserName();public: __property void
11 set_UserName(String*);

12 [VB] Public Property UserName As String

13 [JScript] public function get UserName() : String;public function set
14 UserName(String);

15
16 *Description*

17 The user name associated with the user accessing the URI.

18 Equals

19
20 [C#] public override bool Equals(object rparam);

21 [C++] public: bool Equals(Object* rparam);

22 [VB] Overrides Public Function Equals(ByVal rparam As Object) As Boolean

23 [JScript] public override function Equals(rparam : Object) : Boolean;

24
25 *Description*

1 Compares an existing **System.Uri** instance with the contents of the
2 **System.UriBuilder** for equality.

3 *Return Value:* **true** if the specified **System.Uri** is equal to the **System.Uri**
4 constructed by this **System.UriBuilder** instance; otherwise, **false** .

5 The **System.UriBuilder.Equals(System.Object)** method compares a
6 specified **System.Uri** instance with the **System.Uri** instance built by the
7 **System.UriBuilder** instance. If the two are the same, the
8 **System.UriBuilder.Equals(System.Object)** method returns **true** . The
9 **System.Uri** instance to compare with the current instance.

10 GetHashCode

11
12 [C#] public override int GetHashCode();

13 [C++] public: int GetHashCode();

14 [VB] Overrides Public Function GetHashCode() As Integer

15 [JScript] public override function GetHashCode() : int;

17 Description

18 Returns the hash code for the URI.

19 *Return Value:* The hash code generated for the URI.

20 The hash code is generated without including any fragment included. The
21 URIs <http://www.contoso.com/index.htm#search> and
22 <http://www.contoso.com/index.htm> generate the same hash code.

23 ToString

24
25 [C#] public override string ToString();

1 [C++] public: String* ToString();
2 [VB] Overrides Public Function ToString() As String
3 [JScript] public override function ToString() : String;
4

5 *Description*

6 Returns the display string for the specified **System.UriBuilder** instance.

7 *Return Value:* The string containing the unescaped display name of the
8 **System.UriBuilder** .

9 UriFormatException class (System)

10 ToString

11
12
13 *Description*

14 The exception that is thrown when an invalid Uniform Resource Identifier
15 (URI) is detected.

16 The **System.UriFormatException** is thrown by the Uri class constructor if
17 the supplied URI could not correctly parsed. The format for a valid URI is defined
18 in RFC 2396.

19 UriFormatException

20 *Example Syntax:*

21 ToString

22
23 [C#] public UriFormatException();

24 [C++] public: UriFormatException();

25 [VB] Public Sub New()

[JScript] public function UriFormatException(); Initializes a new instance of the **System.UriFormatException** class.

Description

Initializes a new instance of the **System.UriFormatException** class.

The default constructor initializes a new instance of the **System.UriFormatException** class with all fields set to **null**.

UriFormatException

Example Syntax:

ToString

[C#] public UriFormatException(string textString);

[C++] public: UriFormatException(String* textString);

[VB] Public Sub New(ByVal textString As String)

[JScript] public function UriFormatException(textString : String);

Description

Initializes a new instance of the **System.UriFormatException** class with the specified message.

The UriFormatException constructor initializes the **System.UriFormatException** instance with the **System.Exception.Message** property set to the value of *textString*. The error message string.

UriFormatException

Example Syntax:

ToString

```

1
2 [C#] protected UriFormatException(SerializationInfo serializationInfo,
3 StreamingContext streamingContext);
4 [C++] protected: UriFormatException(SerializationInfo* serializationInfo,
5 StreamingContext streamingContext);
6 [VB] Protected Sub New(ByVal serializationInfo As SerializationInfo, ByVal
7 streamingContext As StreamingContext)
8 [JScript] protected function UriFormatException(serializationInfo :
9 SerializationInfo, streamingContext : StreamingContext);
10     HelpLink
11     HRESULT
12     InnerException
13     Message
14     Source
15     StackTrace
16     TargetSite
17     ISerializable.GetObjectData
18
19 [C#] void ISerializable.GetObjectData(SerializationInfo serializationInfo,
20 StreamingContext streamingContext);
21 [C++] void ISerializable::GetObjectData(SerializationInfo* serializationInfo,
22 StreamingContext streamingContext);
23 [VB] Sub GetObjectData(ByVal serializationInfo As SerializationInfo, ByVal
24 streamingContext As StreamingContext) Implements ISerializable.GetObjectData
25

```

1 [JScript] function ISerializable.GetObjectData(serializationInfo : SerializationInfo,
2 streamingContext : StreamingContext);

3 UriHostNameType enumeration (System)

4 ToString

7 *Description*

8 Defines host name types for the

9 **System.Uri.CheckHostName(System.String)** method.

10 The **System.UriHostNameType** enumeration defines the values that the

11 **System.Uri.CheckHostName(System.String)** method can return.

12 ToString

14 [C#] public const UriHostNameType Basic;

15 [C++] public: const UriHostNameType Basic;

16 [VB] Public Const Basic As UriHostNameType

17 [JScript] public var Basic : UriHostNameType;

19 *Description*

20 The host is set, but the type cannot be determined.

21 ToString

23 [C#] public const UriHostNameType Dns;

24 [C++] public: const UriHostNameType Dns;

25 [VB] Public Const Dns As UriHostNameType

1 [JScript] public var Dns : UriHostNameType;

3 *Description*

4 The host name is a domain name system (DNS) style host name.

5 ToString

7 [C#] public const UriHostNameType IPv4;

8 [C++] public: const UriHostNameType IPv4;

9 [VB] Public Const IPv4 As UriHostNameType

10 [JScript] public var IPv4 : UriHostNameType;

12 *Description*

13 The host name is an Internet Protocol (IP) version 4 host address.

14 ToString

16 [C#] public const UriHostNameType IPv6;

17 [C++] public: const UriHostNameType IPv6;

18 [VB] Public Const IPv6 As UriHostNameType

19 [JScript] public var IPv6 : UriHostNameType;

21 *Description*

22 The host name is an Internet Protocol (IP) version 6 host address.

23 ToString

25 [C#] public const UriHostNameType Unknown;

1 [C++] public: const UriHostNameType Unknown;
2 [VB] Public Const Unknown As UriHostNameType
3 [JScript] public var Unknown : UriHostNameType;

4
5 *Description*

6 The type of the host name is not supplied.
7 UriPartial enumeration (System)
8 ToString

9
10
11 *Description*

12 Defines the parts of a URI for the
13 **System.Uri.GetLeftPart(System.UriPartial)** method.
14 The **System.UriPartial** enumeration defines the values that can be passed
15 to the **System.Uri.GetLeftPart(System.UriPartial)** method.
16 ToString

17
18 [C#] public const UriPartial Authority;
19 [C++] public: const UriPartial Authority;
20 [VB] Public Const Authority As UriPartial
21 [JScript] public var Authority : UriPartial;

22
23 *Description*

24 The scheme and authority segment of the URI.
25 ToString

1
2 [C#] public const UriPartial Path;
3 [C++] public: const UriPartial Path;
4 [VB] Public Const Path As UriPartial
5 [JScript] public var Path : UriPartial;
6

7 *Description*

8 The s

11 SYSTEM.COLLECTIONS NAMESPACE

12 The System.Collections namespace contains classes and interfaces for in-
13 memory data storage and manipulation. There are three main things this
14 namespace is providing. The first is concrete implementations of commonly used
15 collection classes such as a growable array (ArrayList), Hashtable, Stack, Queue,
16 etc. Application developers use these classes as a convenient way to store and
17 retrieve in-memory data. The second set of types in the System.Collections
18 namespace is a set of interfaces to define a formal contract between developers
19 creating new collections and developers consuming collections. For example the
20 IEnumerable interface defines the contract for collections that can offer sequential
21 access via an enumerator. The ASP.NET application model supports data binding
22 controls to data sources that honor the IEnumerable contract. That means that any
23 new collection implementation will automatically be databindable in the ASP.NET
24 model. The third set of types in the System.Collections namespace supports
25 creating strongly typed collections. The CollectionBase class offers data storage

1 that is convenient for developers creating their own, strongly typed collections.
2 This is common for library developers that frequently want to do something such
3 as expose a ControlsCollection that has its Add and Remove method typed to take
4 a Control. By implementing the ControlsCollection to derive from CollectionBase
5 the developer is saved much of the tedious work of rewriting the collection from
6 scratch.

7 The following example shows adding data to a data collection and
8 outputting the data from collection.

```
9
10 static void Main(string[] args)
11 {
12     Console.WriteLine("From an ArrayList");
13     ArrayList l = new ArrayList ();
14     l.Add ("Damien");
15     l.Add ("Mark");
16     l.Add ("Brad");
17     PrintItems (l);
18
19     Console.WriteLine("From a stack");
20     Stack s = new Stack();
21     s.Push(4.5);
22     s.Push(12.3);
23     s.Push(66.2);
24     PrintItems (s);
25
26     Console.WriteLine("From a array");
27     PrintItems (new string[] { "monkey", "cat", "dog" });
28 }
29
30 static void PrintItems (ICollection c)
31 {
32     int ct=0;
33     foreach (object o in c)
34     {
35         Console.WriteLine ("\t{1}:{0}", o,ct++);
36     }
37 }
```

23 The following is a more detailed description of the System.Collections
24 namespace, identifying various classes, interfaces, enumerations, and so forth
25

1 contained in the System.Collections namespace (as well as the
2 System.Collections.Specialized namespace included therein).

3 **System.Collections**

4 The namespace contains interfaces and classes that define various
5 collections of objects, such as lists, queues, bit arrays, hashtables and dictionaries.

7 *Description*

8 The **System.Collections** namespace contains interfaces and classes that
9 define various collections of objects, such as lists, queues, bit arrays, hashtables
10 and dictionaries.

11 **ArrayList** class (System.Collections)

14 *Description*

15 Implements the **System.Collections.ICollection** interface using an array whose
16 size is dynamically increased as required.

17 The capacity of an **System.Collections.ArrayList** is the number of
18 elements the list can hold. As elements are added to an
19 **System.Collections.ArrayList**, the capacity is automatically increased as
20 required through reallocation. The capacity can be decreased by calling
21 **System.Collections.ArrayList.TrimToSize** or by setting the
22 **System.Collections.ArrayList.Capacity** property explicitly.

23 Constructors:

24 **ArrayList**

25 *Example Syntax:*

1 [C#] public ArrayList();

2 [C++] public: ArrayList();

3 [VB] Public Sub New()

4 [JScript] public function ArrayList(); Initializes a new instance of the

5 **System.Collections.ArrayList** class.

6
7
8 *Description*

9 Initializes a new instance of the **System.Collections.ArrayList** class that is
10 empty and has the default initial capacity.

11 The initial capacity is the starting capacity of the new
12 **System.Collections.ArrayList** . The default initial capacity for an
13 **System.Collections.ArrayList** is 16.

14 ArrayList

15 *Example Syntax:*

16
17 [C#] public ArrayList(ICollection c);

18 [C++] public: ArrayList(ICollection* c);

19 [VB] Public Sub New(ByVal c As ICollection)

20 [JScript] public function ArrayList(c : ICollection);

21
22 *Description*

23 Initializes a new instance of the **System.Collections.ArrayList** class that
24 contains elements copied from the specified collection and that has the same initial
25 capacity as the number of elements copied.

The initial capacity is the starting capacity of the new **System.Collections.ArrayList** . If the number of elements added to the list reaches the current capacity, the capacity is automatically doubled. The **System.Collections.ICollection** whose elements are copied to the new list.

ArrayList

Example Syntax:

```
[C#] public ArrayList(int capacity);  
[C++] public: ArrayList(int capacity);  
[VB] Public Sub New(ByVal capacity As Integer)  
[JScript] public function ArrayList(capacity : int);
```

Description

Initializes a new instance of the **System.Collections.ArrayList** class that is empty and has the specified initial capacity.

The initial capacity is the starting capacity of the new **System.Collections.ArrayList** . The default initial capacity for an **System.Collections.ArrayList** is 16. If the specified initial capacity is zero, the default initial capacity is used. The number of elements that the new list is initially capable of storing.

Properties:

Capacity

```
[C#] public virtual int Capacity {get; set;}  
[C++] public: __property virtual int get_Capacity();public: __property virtual void
```

1 set_Capacity(int);

2 [VB] Overridable Public Property Capacity As Integer

3 [JScript] public function get Capacity() : int; public function set Capacity(int);

4
5 *Description*

6 Gets or sets the number of elements that the **System.Collections.ArrayList**
7 can contain.

8 **System.Collections.ArrayList.Capacity** is the number of elements that the
9 **System.Collections.ArrayList** is capable of storing.

10 **Count**

11
12 [C#] public virtual int Count {get;}

13 [C++] public: __property virtual int get_Count();

14 [VB] Overridable Public ReadOnly Property Count As Integer

15 [JScript] public function get Count() : int;

16
17 *Description*

18 Gets the number of elements actually contained in the
19 **System.Collections.ArrayList** .

20 **System.Collections.ArrayList.Count** is the number of elements that are
21 actually in the **System.Collections.ArrayList** .

22 **IsFixedSize**

23
24 [C#] public virtual bool IsFixedSize {get;}

25 [C++] public: __property virtual bool get_IsFixedSize();

1 [VB] Overridable Public ReadOnly Property IsFixedSize As Boolean

2 [JScript] public function get IsFixedSize() : Boolean;

3
4 *Description*

5 Gets a value indicating whether the **System.Collections.ArrayList** has a
6 fixed size.

7 A collection with a fixed size does not allow the addition or removal of
8 elements, but it allows the modification of existing elements.

9 **IsReadOnly**

10
11 [C#] public virtual bool IsReadOnly {get;}

12 [C++] public: __property virtual bool get_IsReadOnly();

13 [VB] Overridable Public ReadOnly Property IsReadOnly As Boolean

14 [JScript] public function get IsReadOnly() : Boolean;

15
16 *Description*

17 Gets a value indicating whether the **System.Collections.ArrayList** is read-
18 only.

19 **IsSynchronized**

20
21 [C#] public virtual bool IsSynchronized {get;}

22 [C++] public: __property virtual bool get_IsSynchronized();

23 [VB] Overridable Public ReadOnly Property IsSynchronized As Boolean

24 [JScript] public function get IsSynchronized() : Boolean;

1
2 *Description*

3 Gets a value indicating whether access to the
4 **System.Collections.ArrayList** is synchronized (thread-safe).

5 To guarantee the thread safety of the **System.Collections.ArrayList** , all
6 operations must be done through the wrapper returned by the
7 **System.Collections.ArrayList.Synchronized(System.Collections.IList)** method.

8 **Item**

9
10 [C#] public virtual object this[int index] {get; set;}

11 [C++] public: __property virtual Object* get_Item(int index);public: __property
12 virtual void set_Item(int index, Object*);

13 [VB] Overridable Public Default Property Item(ByVal index As Integer) As
14 Object

15 [JScript] returnValue = ArrayListObject.Item(index);ArrayListObject.Item(index)
16 = returnValue;

17
18 *Description*

19 Gets or sets the element at the specified index.

20 This property provides the ability to access a specific element in the
21 collection by using the following syntax: myCollection[index] . The zero-based
22 index of the element to get or set.

23 **SyncRoot**

24
25 [C#] public virtual object SyncRoot {get;}

1 [C++] public: __property virtual Object* get_SyncRoot();
2 [VB] Overridable Public ReadOnly Property SyncRoot As Object
3 [JScript] public function get SyncRoot() : Object;

4
5 *Description*

6 Gets an object that can be used to synchronize access to the
7 **System.Collections.ArrayList** .

8 To create a synchronized version of the **System.Collections.ArrayList** ,
9 use the **System.Collections.ArrayList.Synchronized(System.Collections.IList)**
10 method. However, derived classes can provide their own synchronized version of
11 the **System.Collections.ArrayList** using the
12 **System.Collections.ArrayList.SyncRoot** property. The synchronizing code must
13 perform operations on the **System.Collections.ArrayList.SyncRoot** of the
14 **System.Collections.ArrayList** , not directly on the
15 **System.Collections.ArrayList** . This ensures proper operation of collections that
16 are derived from other objects. Specifically, it maintains proper synchronization
17 with other threads that might be simultaneously modifying the
18 **System.Collections.ArrayList** object.

19 Methods:

20 Adapter

21
22 [C#] public static ArrayList Adapter(IList list);
23 [C++] public: static ArrayList* Adapter(IList* list);
24 [VB] Public Shared Function Adapter(ByVal list As IList) As ArrayList
25 [JScript] public static function Adapter(list : IList) : ArrayList;

1
2 *Description*

3 Creates an **System.Collections.ArrayList** wrapper for a specific
4 **System.Collections.IList** .

5 *Return Value:* The **System.Collections.ArrayList** wrapper around the
6 **System.Collections.IList** .

7 **System.Collections.ArrayList.Adapter(System.Collections.IList)** does
8 not copy the contents of **System.Collections.IList** . Instead, it only creates an
9 **System.Collections.ArrayList** wrapper around **System.Collections.IList** ;
10 therefore, changes to the **System.Collections.IList** also affect the
11 **System.Collections.ArrayList** . The **System.Collections.IList** to wrap.

12 **Add**

13
14 [C#] public virtual int Add(object value);

15 [C++] public: virtual int Add(Object* value);

16 [VB] Overridable Public Function Add(ByVal value As Object) As Integer

17 [JScript] public function Add(value : Object) : int;

18
19 *Description*

20 Adds an object to the end of the **System.Collections.ArrayList** .

21 *Return Value:* The **System.Collections.ArrayList** index at which the *value* has
22 been added.

23 If **System.Collections.ArrayList.Count** already equals
24 **System.Collections.ArrayList.Capacity** , the capacity of the list is doubled by
25 automatically reallocating the internal array and copying the existing elements to

the new array before the new element is added. The **System.Object** to be added to the end of the **System.Collections.ArrayList**.

AddRange

[C#] public virtual void AddRange(ICollection c);

[C++] public: virtual void AddRange(ICollection* c);

[VB] Overridable Public Sub AddRange(ByVal c As ICollection)

[JScript] public function AddRange(c : ICollection);

Description

Adds the elements of an **System.Collections.ICollection** to the end of the **System.Collections.ArrayList**.

If the new **System.Collections.ArrayList.Count** (the current **System.Collections.ArrayList.Count** plus the size of the collection) will be greater than **System.Collections.ArrayList.Capacity**, the capacity of the list is either doubled or increased to the new **System.Collections.ArrayList.Count**, whichever is greater. The internal array is automatically reallocated to accommodate the new elements and the existing elements are copied to the new array before the new elements are added. The **System.Collections.ICollection** whose elements should be added to the end of the **System.Collections.ArrayList**.

BinarySearch

[C#] public virtual int BinarySearch(object value);

[C++] public: virtual int BinarySearch(Object* value);

[VB] Overridable Public Function BinarySearch(ByVal value As Object) As

Integer

[JScript] public function BinarySearch(value : Object) : int;

Description

Searches the entire sorted **System.Collections.ArrayList** for an element using the default comparer and returns the zero-based index of the element.

Return Value: The zero-based index of the value in the sorted **System.Collections.ArrayList**, if value is found; otherwise, a negative number, which is the bitwise complement of the index of the next element.

The *value* parameter and each element of the **System.Collections.ArrayList** must implement the **System.IComparable** interface, which is used for comparisons. If the **System.Collections.ArrayList** is not already sorted according to the **System.IComparable** implementation, the result might be incorrect. The **System.Object** to locate.

BinarySearch

[C#] public virtual int BinarySearch(object value, IComparer comparer);

[C++] public: virtual int BinarySearch(Object* value, IComparer* comparer);

[VB] Overridable Public Function BinarySearch(ByVal value As Object, ByVal comparer As IComparer) As Integer

[JScript] public function BinarySearch(value : Object, comparer : IComparer) : int;

Description

Searches the entire sorted **System.Collections.ArrayList** for an element using the specified comparer and returns the zero-based index of the element.

Return Value: The zero-based index of the value in the sorted **System.Collections.ArrayList**, if value is found; otherwise, a negative number, which is the bitwise complement of the index of the next element.

The comparer customizes how the elements are compared. For example, you can use a **System.Collections.CaseInsensitiveComparer** instance as the comparer to perform case-insensitive string searches. The **System.Object** to locate. The **System.Collections.IComparer** implementation to use when comparing elements.

BinarySearch

[C#] public virtual int BinarySearch(int index, int count, object value, IComparer comparer);

[C++] public: virtual int BinarySearch(int index, int count, Object* value, IComparer* comparer);

[VB] Overridable Public Function BinarySearch(ByVal index As Integer, ByVal count As Integer, ByVal value As Object, ByVal comparer As IComparer) As Integer

[JScript] public function BinarySearch(index : int, count : int, value : Object, comparer : IComparer) : int; Uses a binary search algorithm to locate a specific element in the sorted **System.Collections.ArrayList** or a portion of it.

Description

Searches a section of the sorted **System.Collections.ArrayList** for an element using the specified comparer and returns the zero-based index of the element.

Return Value: The zero-based index of the value in the sorted **System.Collections.ArrayList**, if value is found; otherwise, a negative number, which is the bitwise complement of the index of the next element.

The comparer customizes how the elements are compared. For example, you can use a **System.Collections.CaseInsensitiveComparer** instance as the comparer to perform case-insensitive string searches. The zero-based starting index of the range to search. The length of the range to search. The **System.Object** to locate. The **System.Collections.IComparer** implementation to use when comparing elements.

Clear

[C#] public virtual void Clear();

[C++] public: virtual void Clear();

[VB] Overridable Public Sub Clear()

[JScript] public function Clear();

Description

Removes all elements from the **System.Collections.ArrayList**.

System.Collections.ArrayList.Count is set to zero.

Clone

[C#] public virtual object Clone();

1 [C++] public: virtual Object* Clone();

2 [VB] Overridable Public Function Clone() As Object

3 [JScript] public function Clone() : Object;

4
5 *Description*

6 Creates a shallow copy of the **System.Collections.ArrayList** .

7 *Return Value:* A shallow copy of the **System.Collections.ArrayList** .

8 A shallow copy of a collection is a new collection containing references to
9 the same elements as the original collection. The elements themselves or anything
10 referenced by the elements are not copied. In contrast, a deep copy of a collection
11 copies the elements and everything directly or indirectly referenced by the
12 elements.

13 Contains

14
15 [C#] public virtual bool Contains(object item);

16 [C++] public: virtual bool Contains(Object* item);

17 [VB] Overridable Public Function Contains(ByVal item As Object) As Boolean

18 [JScript] public function Contains(item : Object) : Boolean;

19
20 *Description*

21 Determines whether an element is in the **System.Collections.ArrayList** .

22 *Return Value:* **true** if *item* is found in the **System.Collections.ArrayList** ;
23 otherwise, **false** .

24 This method performs a linear search; therefore, the average execution time
25 is proportional to **System.Collections.ArrayList.Count** . That is, this method is

an $O(n)$ operation, where n is **System.Collections.ArrayList.Count**. The **System.Object** to locate in the **System.Collections.ArrayList**. The element to locate can be **null**.

CopyTo

[C#] public virtual void CopyTo(Array array);
[C++] public: virtual void CopyTo(Array* array);
[VB] Overridable Public Sub CopyTo(ByVal array As Array)
[JScript] public function CopyTo(array : Array); Copies the **System.Collections.ArrayList** or a portion of it to a one-dimensional array.

Description

Copies the entire **System.Collections.ArrayList** to a compatible one-dimensional **System.Array**, starting at the beginning of the target array.

The specified array must be of a compatible type. The one-dimensional **System.Array** that is the destination of the elements copied from **System.Collections.ArrayList**. The **System.Array** must have zero-based indexing.

CopyTo

[C#] public virtual void CopyTo(Array array, int arrayIndex);
[C++] public: virtual void CopyTo(Array* array, int arrayIndex);
[VB] Overridable Public Sub CopyTo(ByVal array As Array, ByVal arrayIndex As Integer)
[JScript] public function CopyTo(array : Array, arrayIndex : int);

Description

Copies the entire **System.Collections.ArrayList** to a compatible one-dimensional **System.Array** , starting at the specified index of the target array.

The specified array must be of a compatible type. The one-dimensional **System.Array** that is the destination of the elements copied from **System.Collections.ArrayList**. The **System.Array** must have zero-based indexing. The zero-based index in *array* at which copying begins.

CopyTo

[C#] public virtual void CopyTo(int index, Array array, int arrayIndex, int count);

[C++] public: virtual void CopyTo(int index, Array* array, int arrayIndex, int count);

[VB] Overridable Public Sub CopyTo(ByVal index As Integer, ByVal array As Array, ByVal arrayIndex As Integer, ByVal count As Integer)

[JScript] public function CopyTo(index : int, array : Array, arrayIndex : int, count : int);

Description

Copies a range of elements from the **System.Collections.ArrayList** to a compatible one-dimensional **System.Array** , starting at the specified index of the target array.

The specified array must be of a compatible type. The zero-based index in the source **System.Collections.ArrayList** at which copying begins. The one-dimensional **System.Array** that is the destination of the elements copied from

System.Collections.ArrayList. The **System.Array** must have zero-based indexing. The zero-based index in *array* at which copying begins. The number of elements to copy.

FixedSize

[C#] public static ArrayList FixedSize(ArrayList list);

[C++] public: static ArrayList* FixedSize(ArrayList* list);

[VB] Public Shared Function FixedSize(ByVal list As ArrayList) As ArrayList

[JScript] public static function FixedSize(list : ArrayList) : ArrayList;

Description

Returns an **System.Collections.ArrayList** wrapper with a fixed size.

Return Value: An **System.Collections.ArrayList** wrapper with a fixed size.

This wrapper can be used to prevent additions to and deletions from the original **System.Collections.ArrayList**. The elements can still be modified or replaced. The **System.Collections.ArrayList** to wrap.

FixedSize

[C#] public static IList FixedSize(IList list);

[C++] public: static IList* FixedSize(IList* list);

[VB] Public Shared Function FixedSize(ByVal list As IList) As IList

[JScript] public static function FixedSize(list : IList) : IList; Returns a list wrapper with a fixed size, where elements are allowed to be modified, but not added or removed.

1
2 *Description*

3 Returns an **System.Collections.IList** wrapper with a fixed size.

4 *Return Value:* An **System.Collections.IList** wrapper with a fixed size.

5 This wrapper can be used to prevent additions to and deletions from the
6 original **System.Collections.IList** . The elements can still be modified or replaced.

7 The **System.Collections.IList** to wrap.

8 **GetEnumerator**

9
10 [C#] public virtual IEnumerator GetEnumerator();

11 [C++] public: virtual IEnumerator* GetEnumerator();

12 [VB] Overridable Public Function GetEnumerator() As IEnumerator

13 [JScript] public function GetEnumerator() : IEnumerator; Returns an enumerator
14 that can iterate through the **System.Collections.ArrayList** .

15
16 *Description*

17 Returns an enumerator for the entire **System.Collections.ArrayList** .

18 *Return Value:* An **System.Collections.IEnumerator** for the entire
19 **System.Collections.ArrayList** .

20 Enumerators are intended to be used only to read data in the collection.

21 Enumerators cannot be used to modify the underlying collection.

22 **GetEnumerator**

23
24 [C#] public virtual IEnumerator GetEnumerator(int index, int count);

25 [C++] public: virtual IEnumerator* GetEnumerator(int index, int count);

1 [VB] Overridable Public Function GetEnumerator(ByVal index As Integer, ByVal
2 count As Integer) As IEnumerator

3 [JScript] public function GetEnumerator(index : int, count : int) : IEnumerator;

4
5 *Description*

6 Returns an enumerator for a section of the **System.Collections.ArrayList** .

7 *Return Value:* An **System.Collections.IEnumerator** for the specified section of
8 the **System.Collections.ArrayList** .

9 Enumerators are intended to be used only to read data in the collection.
10 Enumerators cannot be used to modify the underlying collection. The zero-based
11 starting index of the **System.Collections.ArrayList** section that the enumerator
12 should refer to. The number of elements in the **System.Collections.ArrayList**
13 section that the enumerator should refer to.

14 **GetRange**

15
16 [C#] public virtual ArrayList GetRange(int index, int count);

17 [C++] public: virtual ArrayList* GetRange(int index, int count);

18 [VB] Overridable Public Function GetRange(ByVal index As Integer, ByVal
19 count As Integer) As ArrayList

20 [JScript] public function GetRange(index : int, count : int) : ArrayList;

21
22 *Description*

23 Returns an **System.Collections.ArrayList** which represents a subset of the
24 elements in the source **System.Collections.ArrayList** .

Return Value: An **System.Collections.ArrayList** which represents a subset of the elements in the source **System.Collections.ArrayList** .

This method does not create copies of the elements. The new **System.Collections.ArrayList** is only a view window into the source **System.Collections.ArrayList** . However, all subsequent changes to the source **System.Collections.ArrayList** must be done through this view window **System.Collections.ArrayList** . If changes are made directly to the source **System.Collections.ArrayList** , the view window **System.Collections.ArrayList** is invalidated and any operations on it will return an **System.InvalidOperationException** . The zero-based **System.Collections.ArrayList** index at which the range starts. The number of elements in the range.

IndexOf

[C#] public virtual int IndexOf(object value);
[C++] public: virtual int IndexOf(Object* value);
[VB] Overridable Public Function IndexOf(ByVal value As Object) As Integer
[JScript] public function IndexOf(value : Object) : int; Returns the zero-based index of the first occurrence of a value in the **System.Collections.ArrayList** or in a portion of it.

Description

Searches for the specified **System.Object** and returns the zero-based index of the first occurrence within the entire **System.Collections.ArrayList** .

1 *Return Value:* The zero-based index of the first occurrence of *value* within the
2 entire the **System.Collections.ArrayList** , if found; otherwise, -1.

3 The **System.Collections.ArrayList** is searched forward starting at the first
4 element and ending at the last element. The **System.Object** to locate in the
5 **System.Collections.ArrayList**.

6 IndexOf

7
8 [C#] public virtual int IndexOf(object value, int startIndex);

9 [C++] public: virtual int IndexOf(Object* value, int startIndex);

10 [VB] Overridable Public Function IndexOf(ByVal value As Object, ByVal
11 startIndex As Integer) As Integer

12 [JScript] public function IndexOf(value : Object, startIndex : int) : int;

13
14 *Description*

15 Searches for the specified **System.Object** and returns the zero-based index
16 of the first occurrence within the section of the **System.Collections.ArrayList**
17 that extends from the specified index to the last element.

18 *Return Value:* The zero-based index of the first occurrence of *value* within the
19 section of the **System.Collections.ArrayList** that extends from *startIndex* to the
20 last element, if found; otherwise, -1.

21 The **System.Collections.ArrayList** is searched forward starting at
22 *startIndex* and ending at the last element. The **System.Object** to locate in the
23 **System.Collections.ArrayList**. The zero-based starting index of the search.

24 IndexOf
25

1
2 [C#] public virtual int IndexOf(object value, int startIndex, int count);
3 [C++] public: virtual int IndexOf(Object* value, int startIndex, int count);
4 [VB] Overridable Public Function IndexOf(ByVal value As Object, ByVal
5 startIndex As Integer, ByVal count As Integer) As Integer
6 [JScript] public function IndexOf(value : Object, startIndex : int, count : int) : int;

7
8 *Description*

9 Searches for the specified **System.Object** and returns the zero-based index
10 of the first occurrence within the section of the **System.Collections.ArrayList**
11 that starts at the specified index and contains the specified number of elements.

12 *Return Value:* The zero-based index of the first occurrence of *value* within the
13 section of the **System.Collections.ArrayList** that starts at *startIndex* and contains
14 *count* number of elements, if found; otherwise, -1.

15 The **System.Collections.ArrayList** is searched forward starting at
16 *startIndex* and ending at *startIndex* + *count* - 1. The **System.Object** to locate in
17 the **System.Collections.ArrayList**. The zero-based starting index of the search.
18 The number of elements in the section to search.

19 *Insert*

20
21 [C#] public virtual void Insert(int index, object value);
22 [C++] public: virtual void Insert(int index, Object* value);
23 [VB] Overridable Public Sub Insert(ByVal index As Integer, ByVal value As
24 Object)
25 [JScript] public function Insert(index : int, value : Object);

1
2 *Description*

3 Inserts an element into the **System.Collections.ArrayList** at the specified
4 index.

5 If **System.Collections.ArrayList.Count** already equals
6 **System.Collections.ArrayList.Capacity**, the capacity of the list is doubled by
7 automatically reallocating the internal array before the new element is inserted.
8 The zero-based index at which *value* should be inserted. The **System.Object** to
9 insert.

10 **InsertRange**

11
12 [C#] public virtual void InsertRange(int index, ICollection c);

13 [C++] public: virtual void InsertRange(int index, ICollection* c);

14 [VB] Overridable Public Sub InsertRange(ByVal index As Integer, ByVal c As
15 ICollection)

16 [JScript] public function InsertRange(index : int, c : ICollection);
17

18 *Description*

19 Inserts the elements of a collection into the **System.Collections.ArrayList**
20 at the specified index.

21 If the new **System.Collections.ArrayList.Count** (the current
22 **System.Collections.ArrayList.Count** plus the size of the collection) is greater
23 than **System.Collections.ArrayList.Capacity**, the capacity of the list is either
24 doubled or increased to the new count, whichever is greater. The internal array is
25 automatically reallocated to accommodate the new elements. The zero-based index

at which the new elements should be inserted. The

System.Collections.ICollection whose elements should be inserted into the

System.Collections.ArrayList.

LastIndexOf

[C#] public virtual int LastIndexOf(object value);

[C++] public: virtual int LastIndexOf(Object* value);

[VB] Overridable Public Function LastIndexOf(ByVal value As Object) As

Integer

[JScript] public function LastIndexOf(value : Object) : int; Returns the zero-based index of the last occurrence of a value in the **System.Collections.ArrayList** or in a portion of it.

Description

Searches for the specified **System.Object** and returns the zero-based index of the last occurrence within the entire **System.Collections.ArrayList**.

Return Value: The zero-based index of the last occurrence of *value* within the entire the **System.Collections.ArrayList**, if found; otherwise, -1.

The **System.Collections.ArrayList** is searched backward starting at the last element and ending at the first element. The **System.Object** to locate in the **System.Collections.ArrayList**.

LastIndexOf

[C#] public virtual int LastIndexOf(object value, int startIndex);

[C++] public: virtual int LastIndexOf(Object* value, int startIndex);

[VB] Overridable Public Function LastIndexOf(ByVal value As Object, ByVal
startIndex As Integer) As Integer

[JScript] public function LastIndexOf(value : Object, startIndex : int) : int;

Description

Searches for the specified **System.Object** and returns the zero-based index of the last occurrence within the section of the **System.Collections.ArrayList** that extends from the first element to the specified index.

Return Value: The zero-based index of the last occurrence of *value* within the section of the **System.Collections.ArrayList** that extends from the first element to *startIndex* , if found; otherwise, -1.

The **System.Collections.ArrayList** is searched backward starting at *startIndex* and ending at the first element. The **System.Object** to locate in the **System.Collections.ArrayList**. The zero-based starting index of the backward search.

LastIndexOf

[C#] public virtual int LastIndexOf(object value, int startIndex, int count);

[C++] public: virtual int LastIndexOf(Object* value, int startIndex, int count);

[VB] Overridable Public Function LastIndexOf(ByVal value As Object, ByVal
startIndex As Integer, ByVal count As Integer) As Integer

[JScript] public function LastIndexOf(value : Object, startIndex : int, count : int) :
int;

Description

Searches for the specified **System.Object** and returns the zero-based index of the last occurrence within the section of the **System.Collections.ArrayList** that contains the specified number of elements and ends at the specified index.

Return Value: The zero-based index of the last occurrence of *value* within the section of the **System.Collections.ArrayList** that contains *count* number of elements and ends at *startIndex* , if found; otherwise, -1.

The **System.Collections.ArrayList** is searched backward starting at *startIndex* and ending at *startIndex - count + 1*. The **System.Object** to locate in the **System.Collections.ArrayList**. The zero-based starting index of the backward search. The number of elements in the section to search.

ReadOnly

[C#] public static ArrayList ReadOnly(ArrayList list);

[C++] public: static ArrayList* ReadOnly(ArrayList* list);

[VB] Public Shared Function ReadOnly(ByVal list As ArrayList) As ArrayList

[JScript] public static function ReadOnly(list : ArrayList) : ArrayList;

Description

Returns a read-only **System.Collections.ArrayList** wrapper.

Return Value: A read-only **System.Collections.ArrayList** wrapper around *list* .

To prevent any modifications to *list* , expose *list* only through this wrapper.

The **System.Collections.ArrayList** to wrap.

ReadOnly

[C#] public static IList ReadOnly(IList list);

1 [C++] public: static IList* ReadOnly(IList* list);

2 [VB] Public Shared Function ReadOnly(ByVal list As IList) As IList

3 [JScript] public static function ReadOnly(list : IList) : IList; Returns a list wrapper
4 that is read-only.

5
6 *Description*

7 Returns a read-only **System.Collections.IList** wrapper.

8 *Return Value:* A read-only **System.Collections.IList** wrapper around *list* .

9 To prevent any modifications to *list* , expose *list* only through this wrapper.

10 The **System.Collections.IList** to wrap.

11 Remove

12
13 [C#] public virtual void Remove(object obj);

14 [C++] public: virtual void Remove(Object* obj);

15 [VB] Overridable Public Sub Remove(ByVal obj As Object)

16 [JScript] public function Remove(obj : Object);

17
18 *Description*

19 Removes the first occurrence of a specific object from the

20 **System.Collections.ArrayList** .

21 This method performs a linear search; therefore, the average execution time
22 is proportional to **System.Collections.ArrayList.Count** . That is, this method is
23 an $O(n)$ operation, where n is **System.Collections.ArrayList.Count** . The
24 **System.Object** to remove from the **System.Collections.ArrayList**.

25 RemoveAt

1 [C#] public virtual void RemoveAt(int index);

2 [C++] public: virtual void RemoveAt(int index);

3 [VB] Overridable Public Sub RemoveAt(ByVal index As Integer)

4 [JScript] public function RemoveAt(index : int);

5
6
7 *Description*

8 Removes the element at the specified index of the

9 **System.Collections.ArrayList** .

10 In collections such as lists, queues and stacks, the elements that follow the
11 removed element move up to occupy the vacated spot. The zero-based index of the
12 element to remove.

13 **RemoveRange**

14
15 [C#] public virtual void RemoveRange(int index, int count);

16 [C++] public: virtual void RemoveRange(int index, int count);

17 [VB] Overridable Public Sub RemoveRange(ByVal index As Integer, ByVal
18 count As Integer)

19 [JScript] public function RemoveRange(index : int, count : int);

20
21 *Description*

22 Removes a range of elements from the **System.Collections.ArrayList** .

23 In collections such as lists, queues and stacks, the elements that follow the
24 removed elements move up to occupy the vacated spots. The zero-based starting
25 index of the range of elements to remove. The number of elements to remove.

Repeat

```
[C#] public static ArrayList Repeat(object value, int count);  
[C++] public: static ArrayList* Repeat(Object* value, int count);  
[VB] Public Shared Function Repeat(ByVal value As Object, ByVal count As  
Integer) As ArrayList  
[JScript] public static function Repeat(value : Object, count : int) : ArrayList;
```

Description

Returns an **System.Collections.ArrayList** whose elements are copies of the specified value.

Return Value: An **System.Collections.ArrayList** with *count* number of elements, all of which are copies of *value*. The **System.Object** to copy multiple times in the new **System.Collections.ArrayList**. The value to copy can be **null**. The number of times *value* should be copied.

Reverse

```
[C#] public virtual void Reverse();  
[C++] public: virtual void Reverse();  
[VB] Overridable Public Sub Reverse()  
[JScript] public function Reverse(); Reverses the order of the elements in the  
System.Collections.ArrayList or a portion of it.
```

Description

Reverses the order of the elements in the entire

System.Collections.ArrayList .

Reverse

[C#] public virtual void Reverse(int index, int count);

[C++] public: virtual void Reverse(int index, int count);

[VB] Overridable Public Sub Reverse(ByVal index As Integer, ByVal count As Integer)

[JScript] public function Reverse(index : int, count : int);

Description

Reverses the order of the elements in the specified range.

This method uses **System.Array.Reverse(System.Array)** to reverse the order of the elements, such that the element at **System.Collections.ArrayList [i]**, where *i* is any index within the range, moves to **System.Collections.ArrayList [j]**, where *j* equals $index + index + count - i - 1$. The zero-based starting index of the range to reverse. The number of elements in the range to reverse.

SetRange

[C#] public virtual void SetRange(int index, ICollection c);

[C++] public: virtual void SetRange(int index, ICollection* c);

[VB] Overridable Public Sub SetRange(ByVal index As Integer, ByVal c As ICollection)

[JScript] public function SetRange(index : int, c : ICollection);

1
2 *Description*

3 Copies the elements of a collection over a range of elements in the
4 **System.Collections.ArrayList** . The zero-based **System.Collections.ArrayList**
5 index at which to start copying the elements of *c*. The
6 **System.Collections.ICollection** whose elements to copy to the
7 **System.Collections.ArrayList**.

8 Sort

9
10 [C#] public virtual void Sort();
11 [C++] public: virtual void Sort();
12 [VB] Overridable Public Sub Sort()
13 [JScript] public function Sort(); Sorts the elements in the
14 **System.Collections.ArrayList** or a portion of it.
15

16 *Description*

17 Sorts the elements in the entire **System.Collections.ArrayList** using the
18 **System.IComparable** implementation of each element.

19 This method uses **System.Array.Sort(System.Array)** , which uses the
20 QuickSort algorithm. This is an $O(n \log n)$ operation, where *n* is the number of
21 elements to sort.

22 Sort

23
24 [C#] public virtual void Sort(IComparer comparer);
25 [C++] public: virtual void Sort(IComparer* comparer);

1 [VB] Overridable Public Sub Sort(ByVal comparer As IComparer)

2 [JScript] public function Sort(comparer : IComparer);

3
4 *Description*

5 Sorts the elements in the entire **System.Collections.ArrayList** using the
6 specified comparer.

7 This method uses **System.Array.Sort(System.Array)** , which uses the
8 QuickSort algorithm. This is an $O(n \log n)$ operation, where n is the number of
9 elements to sort. The **System.Collections.IComparer** implementation to use
10 when comparing elements.

11 Sort

12
13 [C#] public virtual void Sort(int index, int count, IComparer comparer);

14 [C++] public: virtual void Sort(int index, int count, IComparer* comparer);

15 [VB] Overridable Public Sub Sort(ByVal index As Integer, ByVal count As
16 Integer, ByVal comparer As IComparer)

17 [JScript] public function Sort(index : int, count : int, comparer : IComparer);

18
19 *Description*

20 Sorts the elements in a section of **System.Collections.ArrayList** using the
21 specified comparer.

22 This method uses **System.Array.Sort(System.Array)** , which uses the
23 QuickSort algorithm. This is an $O(n \log n)$ operation, where n is the number of
24 elements to sort. The zero-based starting index of the range to sort. The length of
25

the range to sort. The **System.Collections.IComparer** implementation to use when comparing elements.

Synchronized

[C#] public static ArrayList Synchronized(ArrayList list);

[C++] public: static ArrayList* Synchronized(ArrayList* list);

[VB] Public Shared Function Synchronized(ByVal list As ArrayList) As ArrayList

[JScript] public static function Synchronized(list : ArrayList) : ArrayList;

Description

Returns an **System.Collections.ArrayList** wrapper that is synchronized (thread-safe).

Return Value: An **System.Collections.ArrayList** wrapper that is synchronized (thread-safe).

To guarantee the thread safety of the **System.Collections.ArrayList**, all operations must be done through this wrapper. The **System.Collections.ArrayList** to synchronize.

Synchronized

[C#] public static IList Synchronized(IList list);

[C++] public: static IList* Synchronized(IList* list);

[VB] Public Shared Function Synchronized(ByVal list As IList) As IList

[JScript] public static function Synchronized(list : IList) : IList; Returns a list wrapper that is synchronized (thread-safe).

Description

Returns an **System.Collections.IList** wrapper that is synchronized (thread-safe).

Return Value: An **System.Collections.IList** wrapper that is synchronized (thread-safe).

To guarantee the thread safety of the **System.Collections.ArrayList**, all operations must be done through this wrapper. The **System.Collections.IList** to synchronize.

ToArray

[C#] public virtual object[] ToArray();

[C++] public: virtual Object* ToArray() __gc[];

[VB] Overridable Public Function ToArray() As Object()

[JScript] public function ToArray() : Object[]; Copies the elements of the **System.Collections.ArrayList** to a new array.

Description

Copies the elements of the **System.Collections.ArrayList** to a new **System.Object** array.

Return Value: An **System.Object** array containing copies of the elements of the **System.Collections.ArrayList**.

The elements are copied using **System.Array.Copy(System.Array, System.Array, System.Int32)**, which is an $O(n)$ operation, where n is **System.Collections.ArrayList.Count**.

ToArray

[C#] public virtual Array ToArray(Type type);

[C++] public: virtual Array* ToArray(Type* type);

[VB] Overridable Public Function ToArray(ByVal type As Type) As Array

[JScript] public function ToArray(type : Type) : Array;

Description

Copies the elements of the **System.Collections.ArrayList** to a new array of the specified type.

Return Value: An array of the specified type containing copies of the elements of the **System.Collections.ArrayList**.

The elements are copied using **System.Array.Copy(System.Array, System.Array, System.Int32)**, which is an $O(n)$ operation, where n is **System.Collections.ArrayList.Count**. The **System.Type** of array to create and copy elements to.

TrimToSize

[C#] public virtual void TrimToSize();

[C++] public: virtual void TrimToSize();

[VB] Overridable Public Sub TrimToSize()

[JScript] public function TrimToSize();

Description

1 Sets the capacity to the actual number of elements in the

2 **System.Collections.ArrayList .**

3 This method can be used to minimize a list's memory overhead if no new
4 elements will be added to the list.

5 BitArray class (System.Collections)

6 TrimToSize

7
8
9 *Description*

10 Manages a compact array of bit values, which are represented as Booleans,
11 where **true** indicates that the bit is on (1) and **false** indicates the bit is off (0).

12 The size of a **System.Collections.BitArray** is controlled by the client;
13 indexing past the end of the **System.Collections.BitArray** throws an

14 **System.ArgumentException .**

15 BitArray

16 *Example Syntax:*

17 TrimToSize

18
19 [C#] public BitArray(BitArray bits);

20 [C++] public: BitArray(BitArray* bits);

21 [VB] Public Sub New(ByVal bits As BitArray)

22 [JScript] public function BitArray(bits : BitArray);

23
24 *Description*

1 Initializes a new instance of the **System.Collections.BitArray** class that
2 contains bit values copied from the specified **System.Collections.BitArray** . The
3 **System.Collections.BitArray** to copy.

4 BitArray

5 *Example Syntax:*

6 TrimToSize

7
8 [C#] public BitArray(bool[] values);

9 [C++] public: BitArray(bool values __gc[]);

10 [VB] Public Sub New(ByVal values() As Boolean)

11 [JScript] public function BitArray(values : Boolean[]);

12
13 *Description*

14 Initializes a new instance of the **System.Collections.BitArray** class that
15 contains bit values copied from the specified array of Booleans. An array of
16 Booleans to copy.

17 BitArray

18 *Example Syntax:*

19 TrimToSize

20
21 [C#] public BitArray(byte[] bytes);

22 [C++] public: BitArray(unsigned char bytes __gc[]);

23 [VB] Public Sub New(ByVal bytes() As Byte)

24 [JScript] public function BitArray(bytes : Byte[]);

Description

Initializes a new instance of the **System.Collections.BitArray** class that contains bit values copied from the specified array of bytes.

The first byte in the array represents bits 0 through 7, the second byte represents bits 8 through 15, and so on. The Least Significant Bit of each byte represents the lowest index value: " *bytes* [0] & 1" represents bit 0, " *bytes* [0] & 2" represents bit 1, " *bytes* [0] & 4" represents bit 2, and so on. An array of bytes containing the values to copy, where each byte represents eight consecutive bits.

BitArray

Example Syntax:

TrimToSize

[C#] public BitArray(int length);

[C++] public: BitArray(int length);

[VB] Public Sub New(ByVal length As Integer)

[JScript] public function BitArray(length : int); Initializes a new instance of the **System.Collections.BitArray** class whose capacity and initial values can be specified.

Description

Initializes a new instance of the **System.Collections.BitArray** class that can hold the specified number of bit values, which are initially set to **false** . The number of bit values in the new **System.Collections.BitArray**.

BitArray

Example Syntax:

TrimToSize

[C#] public BitArray(int[] values);

[C++] public: BitArray(int values __gc[]);

[VB] Public Sub New(ByVal values() As Integer)

[JScript] public function BitArray(values : int[]);

Description

Initializes a new instance of the **System.Collections.BitArray** class that contains bit values copied from the specified array of 32-bit integers.

The number in the first *values* array element represents bits 0 through 31, the second number in the array represents bits 32 through 63, and so on. The Least Significant Bit of each integer represents the lowest index value: "*values* [0] & 1" represents bit 0, "*values* [0] & 2" represents bit 1, "*values* [0] & 4" represents bit 2, and so on. An array of integers containing the values to copy, where each integer represents 32 consecutive bits.

BitArray

Example Syntax:

TrimToSize

[C#] public BitArray(int length, bool defaultValue);

[C++] public: BitArray(int length, bool defaultValue);

[VB] Public Sub New(ByVal length As Integer, ByVal defaultValue As Boolean)

[JScript] public function BitArray(length : int, defaultValue : Boolean);

Description

Initializes a new instance of the **System.Collections.BitArray** class that can hold the specified number of bit values, which are initially set to the specified value. The number of bit values in the new **System.Collections.BitArray**. The Boolean value to assign to each bit.

Count

TrimToSize

[C#] public int Count {get;}

[C++] public: __property int get_Count();

[VB] Public ReadOnly Property Count As Integer

[JScript] public function get Count() : int;

Description

Gets the number of elements contained in the **System.Collections.BitArray**.
System.Collections.BitArray.Length and **System.Collections.BitArray.Count** return the same value.

IsReadOnly

TrimToSize

[C#] public bool IsReadOnly {get;}

[C++] public: __property bool get_IsReadOnly();

[VB] Public ReadOnly Property IsReadOnly As Boolean

1 [JScript] public function get IsReadOnly() : Boolean;

3 *Description*

4 Gets a value indicating whether the **System.Collections.BitArray** is read-
5 only.

6 **System.Collections.BitArray** implements the
7 **System.Collections.BitArray.IsReadOnly** property because it is required by the
8 **System.Collections.IList** interface.

9 IsSynchronized

10 TrimToSize

12 [C#] public bool IsSynchronized {get;}

13 [C++] public: __property bool get_IsSynchronized();

14 [VB] Public ReadOnly Property IsSynchronized As Boolean

15 [JScript] public function get IsSynchronized() : Boolean;

17 *Description*

18 Gets a value indicating whether access to the **System.Collections.BitArray**
19 is synchronized (thread-safe).

20 **System.Collections.BitArray** implements the
21 **System.Collections.BitArray.IsSynchronized** property because it is required by
22 the **System.Collections.ICollection** interface.

23 Item

24 TrimToSize

1 [C#] public bool this[int index] {get; set;}

2 [C++] public: __property bool get_Item(int index);public: __property void
3 set_Item(int index, bool);

4 [VB] Public Default Property Item(ByVal index As Integer) As Boolean

5 [JScript] returnValue = BitArrayObject.Item(index);BitArrayObject.Item(index) =
6 returnValue;

7
8
9 *Description*

10 Gets or sets the value of the bit at a specific position in the

11 **System.Collections.BitArray** .

12 This property provides the ability to access a specific element in the
13 collection by using the following syntax: myCollection[index] . The zero-based
14 index of the value to get or set.

15 Length

16 TrimToSize

17
18 [C#] public int Length {get; set;}

19 [C++] public: __property int get_Length();public: __property void
20 set_Length(int);

21 [VB] Public Property Length As Integer

22 [JScript] public function get Length() : int;public function set Length(int);

23
24 *Description*

25 Gets or sets the number of elements in the **System.Collections.BitArray** .

System.Collections.BitArray.Length and

System.Collections.BitArray.Count return the same value.

SyncRoot

TrimToSize

[C#] public object SyncRoot {get;}

[C++] public: __property Object* get_SyncRoot();

[VB] Public ReadOnly Property SyncRoot As Object

[JScript] public function get SyncRoot() : Object;

Description

Gets an object that can be used to synchronize access to the

System.Collections.BitArray .

Derived classes can provide their own synchronized version of the

System.Collections.BitArray using the **System.Collections.BitArray.SyncRoot**

property. The synchronizing code must perform operations on the

System.Collections.BitArray.SyncRoot of the **System.Collections.BitArray** ,

not directly on the **System.Collections.BitArray** . This ensures proper operation

of collections that are derived from other objects. Specifically, it maintains proper

synchronization with other threads that might be simultaneously modifying the

System.Collections.BitArray object.

And

[C#] public BitArray And(BitArray value);

[C++] public: BitArray* And(BitArray* value);

1 [VB] Public Function And(ByVal value As BitArray) As BitArray

2 [JScript] public function And(value : BitArray) : BitArray;

3
4 *Description*

5 Performs the bitwise AND operation on the elements in the current
6 **System.Collections.BitArray** against the corresponding elements in the specified
7 **System.Collections.BitArray** .

8 *Return Value:* A **System.Collections.BitArray** containing the result of the bitwise
9 AND operation on the elements in the current **System.Collections.BitArray**
10 against the corresponding elements in the specified **System.Collections.BitArray**
11 .

12 The bitwise AND operation returns **true** if both operands are **true** , and
13 returns **false** if one or both operands are **false** . The **System.Collections.BitArray**
14 with which to perform the bitwise AND operation.

15 **Clone**

16
17 [C#] public object Clone();

18 [C++] public: __sealed Object* Clone();

19 [VB] NotOverridable Public Function Clone() As Object

20 [JScript] public function Clone() : Object;

21
22 *Description*

23 Creates a shallow copy of the **System.Collections.BitArray** .

24 *Return Value:* A shallow copy of the **System.Collections.BitArray** .
25

A shallow copy of a collection is a new collection containing references to the same elements as the original collection. The elements themselves or anything referenced by the elements are not copied. In contrast, a deep copy of a collection copies the elements and everything directly or indirectly referenced by the elements.

CopyTo

[C#] public void CopyTo(Array array, int index);

[C++] public: __sealed void CopyTo(Array* array, int index);

[VB] NotOverridable Public Sub CopyTo(ByVal array As Array, ByVal index As Integer)

[JScript] public function CopyTo(array : Array, index : int);

Description

Copies the entire **System.Collections.BitArray** to a compatible one-dimensional **System.Array**, starting at the specified index of the target array.

The specified array must be of a compatible type. Only **bool**, **int** and **byte** types of arrays are supported. The one-dimensional **System.Array** that is the destination of the elements copied from **System.Collections.BitArray**. The **System.Array** must have zero-based indexing. The zero-based index in *array* at which copying begins.

Get

[C#] public bool Get(int index);

[C++] public: bool Get(int index);

1 [VB] Public Function Get(ByVal index As Integer) As Boolean

2 [JScript] public function Get(index : int) : Boolean;

3
4 *Description*

5 Gets the value of the bit at a specific position in the

6 **System.Collections.BitArray** .

7 *Return Value:* The value of the bit at position *index* . The zero-based index of the
8 value to get.

9 GetEnumerator

10
11 [C#] public IEnumerator GetEnumerator();

12 [C++] public: __sealed IEnumerator* GetEnumerator();

13 [VB] NotOverridable Public Function GetEnumerator() As IEnumerator

14 [JScript] public function GetEnumerator() : IEnumerator;

15
16 *Description*

17 Returns an enumerator that can iterate through the

18 **System.Collections.BitArray** .

19 *Return Value:* An **System.Collections.IEnumerator** for the entire

20 **System.Collections.BitArray** .

21 Enumerators are intended to be used only to read data in the collection.

22 Enumerators cannot be used to modify the underlying collection.

23 Not

24
25 [C#] public BitArray Not();


```

1 [C++] public: BitArray* Not();
2 [VB] Public Function Not() As BitArray
3 [JScript] public function Not() : BitArray;

```

5 *Description*

6 Inverts all the bit values in the current **System.Collections.BitArray** , so
7 that elements set to **true** are changed to **false** , and elements set to **false** are
8 changed to **true** .

9 *Return Value:* The current instance with inverted bit values.

10 Or

```

11
12 [C#] public BitArray Or(BitArray value);
13 [C++] public: BitArray* Or(BitArray* value);
14 [VB] Public Function Or(ByVal value As BitArray) As BitArray
15 [JScript] public function Or(value : BitArray) : BitArray;

```

17 *Description*

18 Performs the bitwise OR operation on the elements in the current
19 **System.Collections.BitArray** against the corresponding elements in the specified
20 **System.Collections.BitArray** .

21 *Return Value:* A **System.Collections.BitArray** containing the result of the bitwise
22 OR operation on the elements in the current **System.Collections.BitArray** against
23 the corresponding elements in the specified **System.Collections.BitArray** .

The bitwise OR operation returns **true** if one or both operands are **true** , and returns **false** if both operands are **false** . The **System.Collections.BitArray** with which to perform the bitwise OR operation.

Set

[C#] public void Set(int index, bool value);

[C++] public: void Set(int index, bool value);

[VB] Public Sub Set(ByVal index As Integer, ByVal value As Boolean)

[JScript] public function Set(index : int, value : Boolean);

Description

Sets the bit at a specific position in the **System.Collections.BitArray** to the specified value. The zero-based index of the bit to set. The Boolean value to assign to the bit.

SetAll

[C#] public void SetAll(bool value);

[C++] public: void SetAll(bool value);

[VB] Public Sub SetAll(ByVal value As Boolean)

[JScript] public function SetAll(value : Boolean);

Description

Sets all bits in the **System.Collections.BitArray** to the specified value. The Boolean value to assign to all bits.

Xor

```

1
2 [C#] public BitArray Xor(BitArray value);
3 [C++] public: BitArray* Xor(BitArray* value);
4 [VB] Public Function Xor(ByVal value As BitArray) As BitArray
5 [JScript] public function Xor(value : BitArray) : BitArray;
6

```

Description

Performs the bitwise eXclusive OR operation on the elements in the current **System.Collections.BitArray** against the corresponding elements in the specified **System.Collections.BitArray** .

Return Value: A **System.Collections.BitArray** containing the result of the bitwise eXclusive OR operation on the elements in the current **System.Collections.BitArray** against the corresponding elements in the specified **System.Collections.BitArray** .

The bitwise exclusive OR operation returns **true** if exactly one operand is **true** , and returns **false** if both operands have the same Boolean value. The **System.Collections.BitArray** with which to perform the bitwise eXclusive OR operation.

CaseInsensitiveComparer class (System.Collections)

Xor

Description

Compares two objects for equivalence, ignoring the case of strings.

System.Collections.CaseInsensitiveComparer implements the **System.Collections.IComparer** interface supporting case-insensitive comparisons on strings, just as **System.Collections.CaseInsensitiveHashCodeProvider** implements the **System.Collections.IHashCodeProvider** interface supporting case-insensitive comparisons on strings.

CaseInsensitiveComparer

Example Syntax:

Xor

[C#] public CaseInsensitiveComparer();

[C++] public: CaseInsensitiveComparer();

[VB] Public Sub New()

[JScript] public function CaseInsensitiveComparer(); Initializes a new instance of the **System.Collections.CaseInsensitiveComparer** class.

Description

Initializes a new instance of the **System.Collections.CaseInsensitiveComparer** class using the **System.Threading.Thread.CurrentCulture** of the current thread.

Comparison procedures use the **System.Threading.Thread.CurrentCulture** of the current thread to determine the sort order. String comparisons might have different results depending on the culture. For more information on culture-specific comparisons, see the **System.Globalization** namespace and .

CaseInsensitiveComparer

Example Syntax:

Xor

[C#] public CaseInsensitiveComparer(CultureInfo culture);

[C++] public: CaseInsensitiveComparer(CultureInfo* culture);

[VB] Public Sub New(ByVal culture As CultureInfo)

[JScript] public function CaseInsensitiveComparer(culture : CultureInfo);

Description

Initializes a new instance of the

System.Collections.CaseInsensitiveComparer class using the specified

System.Globalization.CultureInfo .

Comparison procedures use the specified

System.Globalization.CultureInfo to determine the sort order. String

comparisons might have different results depending on the culture. For more

information on culture-specific comparisons, see the **System.Globalization**

namespace and . The **System.Globalization.CultureInfo** to use for the new

System.Collections.CaseInsensitiveComparer.

Default

Xor

[C#] public static CaseInsensitiveComparer Default {get;}

[C++] public: __property static CaseInsensitiveComparer* get_Default();

[VB] Public Shared ReadOnly Property Default As CaseInsensitiveComparer

1 [JScript] public static function get Default() : CaseInsensitiveComparer;

2
3 *Description*

4 Gets an instance of **System.Collections.CaseInsensitiveComparer** that is
5 always available.

6 Compare

7
8 [C#] public int Compare(object a, object b);

9 [C++] public: __sealed int Compare(Object* a, Object* b);

10 [VB] NotOverridable Public Function Compare(ByVal a As Object, ByVal b As
11 Object) As Integer

12 [JScript] public function Compare(a : Object, b : Object) : int;

13
14 *Description*

15 Performs a case-insensitive comparison of two objects of the same type and
16 returns a value indicating whether one is less than, equal to or greater than the
17 other.

18 *Return Value:* Value Condition Less than zero *a* is less than *b* , with casing
19 ignored.

20 If *a* implements **System.IComparable** , then *a* . The first object to
21 compare. The second object to compare.

22 CaseInsensitiveHashCodeProvider class (System.Collections)

23 ToString

1
2
3 *Description*

4 Supplies a hash code for an object, using a hashing algorithm that ignores
5 the case of strings.

6 **System.Collections.CaseInsensitiveHashCodeProvider** implements the
7 **System.Collections.IHashCodeProvider** interface supporting case-insensitive
8 comparisons on strings, just as **System.Collections.CaseInsensitiveComparer**
9 implements the **System.Collections.IComparer** interface supporting case-
10 insensitive comparisons on strings.

11 CaseInsensitiveHashCodeProvider

12 *Example Syntax:*

13 ToString

14
15 [C#] public CaseInsensitiveHashCodeProvider();

16 [C++] public: CaseInsensitiveHashCodeProvider();

17 [VB] Public Sub New()

18 [JScript] public function CaseInsensitiveHashCodeProvider(); Initializes a new
19 instance of the **System.Collections.CaseInsensitiveHashCodeProvider** class.

20
21 *Description*

22 Initializes a new instance of the
23 **System.Collections.CaseInsensitiveHashCodeProvider** class using the current
24 **System.Globalization.CultureInfo** .

25 The **System.Globalization.CultureInfo** provides information about casing.

CaseInsensitiveHashCodeProvider

Example Syntax:

ToString

[C#] public CaseInsensitiveHashCodeProvider(CultureInfo culture);

[C++] public: CaseInsensitiveHashCodeProvider(CultureInfo* culture);

[VB] Public Sub New(ByVal culture As CultureInfo)

[JScript] public function CaseInsensitiveHashCodeProvider(culture : CultureInfo);

Description

Initializes a new instance of the **System.Collections.CaseInsensitiveHashCodeProvider** class using the current **System.Globalization.CultureInfo**.

The **System.Globalization.CultureInfo** provides information about casing. The **System.Globalization.CultureInfo** to use for the new **System.Collections.CaseInsensitiveHashCodeProvider**.

Default

ToString

[C#] public static CaseInsensitiveHashCodeProvider Default {get;}

[C++] public: __property static CaseInsensitiveHashCodeProvider* get_Default();

[VB] Public Shared ReadOnly Property Default As

CaseInsensitiveHashCodeProvider

[JScript] public static function get Default() : CaseInsensitiveHashCodeProvider;

Description

Gets an instance of **System.Collections.CaseInsensitiveHashCodeProvider** that is always available.

GetHashCode

[C#] public int GetHashCode(object obj);

[C++] public: __sealed int GetHashCode(Object* obj);

[VB] NotOverridable Public Function GetHashCode(ByVal obj As Object) As

Integer

[JScript] public function GetHashCode(obj : Object) : int; Returns a hash code, using a hashing algorithm that ignores the case of strings.

Description

Returns a hash code for the given object, using a hashing algorithm that ignores the case of strings.

Return Value: A hash code for the given object, using a hashing algorithm that ignores the case of strings.

The return value from this method must not be persisted for two reasons. First, the hash function of a class might be altered to generate a better distribution, rendering any values from the old hash function useless. Second, the default implementation of this class does not guarantee that the same value will be returned by different instances. The **System.Object** for which a hash code is to be returned.

CollectionBase class (System.Collections)

ToString

Description

Provides the **abstract** base class for a strongly typed collection.

A **System.Collections.CollectionBase** instance is always modifiable. See **System.Collections.ReadOnlyCollectionBase** for a read-only version of this class.

CollectionBase

Example Syntax:

ToString

[C#] protected CollectionBase();

[C++] protected: CollectionBase();

[VB] Protected Sub New()

[JScript] protected function CollectionBase();

Count

ToString

[C#] public int Count {get;}

[C++] public: __property int get_Count();

[VB] Public ReadOnly Property Count As Integer

[JScript] public function get Count() : int;

Description

1 Gets the number of elements contained in the
2 **System.Collections.CollectionBase** instance.

3 InnerList

4 ToString

5
6 [C#] protected ArrayList InnerList {get;}

7 [C++] protected: __property ArrayList* get_InnerList();

8 [VB] Protected ReadOnly Property InnerList As ArrayList

9 [JScript] protected function get InnerList() : ArrayList;

10
11 *Description*

12 Gets an **System.Collections.ArrayList** containing the list of elements in
13 the **System.Collections.CollectionBase** instance.

14 List

15 ToString

16
17 [C#] protected IList List {get;}

18 [C++] protected: __property IList* get_List();

19 [VB] Protected ReadOnly Property List As IList

20 [JScript] protected function get List() : IList;

21
22 *Description*

23 Gets an **System.Collections.IList** containing the list of elements in the
24 **System.Collections.CollectionBase** instance.

25 Clear

```

1
2 [C#] public void Clear();
3 [C++] public: __sealed void Clear();
4 [VB] NotOverridable Public Sub Clear()
5 [JScript] public function Clear();
6

```

7 *Description*

8 Removes all objects from the **System.Collections.CollectionBase** instance.

9 **System.Collections.CollectionBase.Count** is set to zero.

10 GetEnumerator

```

11
12 [C#] public IEnumerator GetEnumerator();
13 [C++] public: __sealed IEnumerator* GetEnumerator();
14 [VB] NotOverridable Public Function GetEnumerator() As IEnumerator
15 [JScript] public function GetEnumerator() : IEnumerator;
16

```

17 *Description*

18 Returns an enumerator that can iterate through the

19 **System.Collections.CollectionBase** instance.

20 *Return Value:* An **System.Collections.IEnumerator** for the

21 **System.Collections.CollectionBase** instance.

22 Enumerators are intended to be used only to read data in the collection.

23 Enumerators cannot be used to modify the underlying collection.

24 OnClear

1
2 [C#] protected virtual void OnClear();

3 [C++] protected: virtual void OnClear();

4 [VB] Overridable Protected Sub OnClear()

5 [JScript] protected function OnClear();

6
7 *Description*

8 Performs additional custom processes when clearing the contents of the
9 **System.Collections.CollectionBase** instance.

10 The default implementation of this method is intended to be overridden by
11 a derived class to perform some action before the collection is cleared.

12 OnClearComplete

13
14 [C#] protected virtual void OnClearComplete();

15 [C++] protected: virtual void OnClearComplete();

16 [VB] Overridable Protected Sub OnClearComplete()

17 [JScript] protected function OnClearComplete();

18
19 *Description*

20 Performs additional custom processes after clearing the contents of the
21 **System.Collections.CollectionBase** instance.

22 The default implementation of this method is intended to be overridden by
23 a derived class to perform some action after the collection is cleared.

24 OnInsert

1
2 [C#] protected virtual void OnInsert(int index, object value);

3 [C++] protected: virtual void OnInsert(int index, Object* value);

4 [VB] Overridable Protected Sub OnInsert(ByVal index As Integer, ByVal value
5 As Object)

6 [JScript] protected function OnInsert(index : int, value : Object);

7
8 *Description*

9 Performs additional custom processes before inserting a new element into
10 the **System.Collections.CollectionBase** instance.

11 The default implementation of this method is intended to be overridden by
12 a derived class to perform some action before the specified element is inserted.

13 The zero-based index at which to insert *value*. The new value of the element at
14 *index*.

15 **OnInsertComplete**

16
17 [C#] protected virtual void OnInsertComplete(int index, object value);

18 [C++] protected: virtual void OnInsertComplete(int index, Object* value);

19 [VB] Overridable Protected Sub OnInsertComplete(ByVal index As Integer,
20 ByVal value As Object)

21 [JScript] protected function OnInsertComplete(index : int, value : Object);

22
23 *Description*

24 Performs additional custom processes after inserting a new element into the
25 **System.Collections.CollectionBase** instance.

The default implementation of this method is intended to be overridden by a derived class to perform some action after the specified element is inserted. The zero-based index at which to insert *value*. The new value of the element at *index*.

OnRemove

[C#] protected virtual void OnRemove(int index, object value);

[C++] protected: virtual void OnRemove(int index, Object* value);

[VB] Overridable Protected Sub OnRemove(ByVal index As Integer, ByVal value As Object)

[JScript] protected function OnRemove(index : int, value : Object);

Description

Performs additional custom processes when removing an element from the **System.Collections.CollectionBase** instance.

The default implementation of this method is intended to be overridden by a derived class to perform some action before the specified element is removed. The zero-based index at which *value* can be found. The value of the element to remove from *index*.

OnRemoveComplete

[C#] protected virtual void OnRemoveComplete(int index, object value);

[C++] protected: virtual void OnRemoveComplete(int index, Object* value);

[VB] Overridable Protected Sub OnRemoveComplete(ByVal index As Integer, ByVal value As Object)

[JScript] protected function OnRemoveComplete(index : int, value : Object);

Description

Performs additional custom processes after removing an element from the **System.Collections.CollectionBase** instance.

The default implementation of this method is intended to be overridden by a derived class to perform some action after the specified element is removed. The zero-based index at which *value* can be found. The value of the element to remove from *index*.

OnSet

[C#] protected virtual void OnSet(int index, object oldValue, object newValue);

[C++] protected: virtual void OnSet(int index, Object* oldValue, Object* newValue);

[VB] Overridable Protected Sub OnSet(ByVal index As Integer, ByVal oldValue As Object, ByVal newValue As Object)

[JScript] protected function OnSet(index : int, oldValue : Object, newValue : Object);

Description

Performs additional custom processes before setting a value in the **System.Collections.CollectionBase** instance.

The default implementation of this method is intended to be overridden by a derived class to perform some action before the specified element is set. The zero-based index at which *oldValue* can be found. The value to replace with *newValue*. The new value of the element at *index*.

OnSetComplete

[C#] protected virtual void OnSetComplete(int index, object oldValue, object newValue);

[C++] protected: virtual void OnSetComplete(int index, Object* oldValue, Object* newValue);

[VB] Overridable Protected Sub OnSetComplete(ByVal index As Integer, ByVal oldValue As Object, ByVal newValue As Object)

[JScript] protected function OnSetComplete(index : int, oldValue : Object, newValue : Object);

Description

Performs additional custom processes after setting a value in the **System.Collections.CollectionBase** instance.

The default implementation of this method is intended to be overridden by a derived class to perform some action after the specified element is set. The zero-based index at which *oldValue* can be found. The value to replace with *newValue*. The new value of the element at *index*.

OnValidate

[C#] protected virtual void OnValidate(object value);

[C++] protected: virtual void OnValidate(Object* value);

[VB] Overridable Protected Sub OnValidate(ByVal value As Object)

[JScript] protected function OnValidate(value : Object);

Description

Performs additional custom processes when validating a value.

The default implementation of this method determines whether *value* is **null**, and, if so, throws **System.ArgumentNullException**. It is intended to be overridden by a derived class to perform additional action when the specified element is validated. The object to validate.

RemoveAt

[C#] public void RemoveAt(int index);

[C++] public: __sealed void RemoveAt(int index);

[VB] NotOverridable Public Sub RemoveAt(ByVal index As Integer)

[JScript] public function RemoveAt(index : int);

Description

Removes the element at the specified index of the **System.Collections.CollectionBase** instance.

In collections such as lists, queues and stacks, the elements that follow the removed element move up to occupy the vacated spot. The zero-based index of the element to remove.

ICollection.CopyTo

[C#] void ICollection.CopyTo(Array array, int index);

[C++] void ICollection::CopyTo(Array* array, int index);

[VB] Sub CopyTo(ByVal array As Array, ByVal index As Integer) Implements

```

1  ICollection.CopyTo
2  [JScript] function ICollection.CopyTo(array : Array, index : int);
3      IList.Add
4
5  [C#] int IList.Add(object value);
6  [C++] int IList::Add(Object* value);
7  [VB] Function Add(ByVal value As Object) As Integer Implements IList.Add
8  [JScript] function IList.Add(value : Object) : int;
9      IList.Contains
10
11 [C#] bool IList.Contains(object value);
12 [C++] bool IList::Contains(Object* value);
13 [VB] Function Contains(ByVal value As Object) As Boolean Implements
14     IList.Contains
15 [JScript] function IList.Contains(value : Object) : Boolean;
16     IList.IndexOf
17
18 [C#] int IList.IndexOf(object value);
19 [C++] int IList::IndexOf(Object* value);
20 [VB] Function IndexOf(ByVal value As Object) As Integer Implements
21     IList.IndexOf
22 [JScript] function IList.IndexOf(value : Object) : int;
23     IList.Insert
24
25 [C#] void IList.Insert(int index, object value);

```

```

1 [C++] void IList::Insert(int index, Object* value);
2 [VB] Sub Insert(ByVal index As Integer, ByVal value As Object) Implements
3 IList.Insert
4 [JScript] function IList.Insert(index : int, value : Object);
5     IList.Remove
6

```

```

7 [C#] void IList.Remove(object value);
8 [C++] void IList::Remove(Object* value);
9 [VB] Sub Remove(ByVal value As Object) Implements IList.Remove
10 [JScript] function IList.Remove(value : Object);
11     Comparer class (System.Collections)
12     ToString
13
14

```

Description

Compares two objects for equivalence, where string comparisons are case-sensitive.

This class is the default implementation of the **System.Collections.IComparer** interface. The **System.Collections.CaseInsensitiveComparer** class is the implementation of the **System.Collections.IComparer** interface that performs case-insensitive string comparisons.

ToString

```

25 [C#] public static readonly Comparer Default;

```

1 [C++] public: static Comparer* Default;

2 [VB] Public Shared ReadOnly Default As Comparer

3 [JScript] public static var Default : Comparer;

4
5 *Description*

6 Gets an instance of **System.Collections.Comparer** that is always available.

7 An instance of **System.Collections.Comparer** that is always available.

8 Compare

9
10 [C#] public int Compare(object a, object b);

11 [C++] public: __sealed int Compare(Object* a, Object* b);

12 [VB] NotOverridable Public Function Compare(ByVal a As Object, ByVal b As

13 Object) As Integer

14 [JScript] public function Compare(a : Object, b : Object) : int;

15
16 *Description*

17 Performs a case-sensitive comparison of two objects of the same type and
18 returns a value indicating whether one is less than, equal to or greater than the
19 other.

20 *Return Value:* Value Condition Less than zero *a* is less than *b* .

21 If *a* implements **System.IComparable** , then *a* . The first object to
22 compare. The second object to compare.

23 DictionaryBase class (System.Collections)

24 ToString

1
2
3 *Description*

4 Provides the **abstract** base class for a strongly typed collection of key-and-
5 value pairs.

6 Each element is a key-and-value pair stored in a
7 **System.Collections.DictionaryEntry** object.

8 DictionaryBase

9 *Example Syntax:*

10 ToString

11
12 [C#] protected DictionaryBase();

13 [C++] protected: DictionaryBase();

14 [VB] Protected Sub New()

15 [JScript] protected function DictionaryBase();

16 Count

17 ToString

18
19 [C#] public int Count {get;}

20 [C++] public: __property int get_Count();

21 [VB] Public ReadOnly Property Count As Integer

22 [JScript] public function get Count() : int;

23
24 *Description*
25

Gets the number of elements contained in the
System.Collections.DictionaryBase instance.

Dictionary

ToString

[C#] protected IDictionary Dictionary {get;}

[C++] protected: __property IDictionary* get_Dictionary();

[VB] Protected ReadOnly Property Dictionary As IDictionary

[JScript] protected function get Dictionary() : IDictionary;

Description

Gets the list of elements contained in the
System.Collections.DictionaryBase instance.

InnerHashtable

ToString

[C#] protected Hashtable InnerHashtable {get;}

[C++] protected: __property Hashtable* get_InnerHashtable();

[VB] Protected ReadOnly Property InnerHashtable As Hashtable

[JScript] protected function get InnerHashtable() : Hashtable;

Description

Gets the list of elements contained in the
System.Collections.DictionaryBase instance.

Clear

```

1
2 [C#] public void Clear();
3 [C++] public: __sealed void Clear();
4 [VB] NotOverridable Public Sub Clear()
5 [JScript] public function Clear();
6

```

7 *Description*

8 Clears the contents of the **System.Collections.DictionaryBase** instance.
9 **System.Collections.DictionaryBase.Count** is set to zero.

10 *CopyTo*

```

11
12 [C#] public void CopyTo(Array array, int index);
13 [C++] public: __sealed void CopyTo(Array* array, int index);
14 [VB] NotOverridable Public Sub CopyTo(ByVal array As Array, ByVal index As
15 Integer)
16 [JScript] public function CopyTo(array : Array, index : int);
17

```

18 *Description*

19 Copies the **System.Collections.DictionaryBase** elements to a one-
20 dimensional **System.Array** at the specified index.

21 The elements are copied to the **System.Array** in the same order in which
22 the enumerator iterates through the **System.Collections.DictionaryBase** . The
23 one-dimensional **System.Array** that is the destination of the
24 **System.Collections.DictionaryEntry** objects copied from the
25

System.Collections.DictionaryBase instance. The **System.Array** must have zero-based indexing. The zero-based index in *array* at which copying begins.

GetEnumerator

[C#] public IDictionaryEnumerator GetEnumerator();

[C++] public: __sealed IDictionaryEnumerator* GetEnumerator();

[VB] NotOverridable Public Function GetEnumerator() As IDictionaryEnumerator

[JScript] public function GetEnumerator() : IDictionaryEnumerator;

Description

Returns an enumerator that can iterate through the **System.Collections.DictionaryBase** instance.

Return Value: An **System.Collections.IEnumerator** for the **System.Collections.DictionaryBase** instance.

Enumerators are intended to be used only to read data in the collection. Enumerators cannot be used to modify the underlying collection.

OnClear

[C#] protected virtual void OnClear();

[C++] protected: virtual void OnClear();

[VB] Overridable Protected Sub OnClear()

[JScript] protected function OnClear();

Description

1 Performs additional custom processes before clearing the contents of the
2 **System.Collections.DictionaryBase** instance.

3 The default implementation of this method is intended to be overridden by
4 a derived class to perform some action before the collection is cleared.

5 OnClearComplete

6
7 [C#] protected virtual void OnClearComplete();

8 [C++] protected: virtual void OnClearComplete();

9 [VB] Overridable Protected Sub OnClearComplete()

10 [JScript] protected function OnClearComplete();

11 12 *Description*

13 Performs additional custom processes after clearing the contents of the
14 **System.Collections.DictionaryBase** instance.

15 The default implementation of this method is intended to be overridden by
16 a derived class to perform some action after the collection is cleared.

17 OnGet

18
19 [C#] protected virtual object OnGet(object key, object currentValue);

20 [C++] protected: virtual Object* OnGet(Object* key, Object* currentValue);

21 [VB] Overridable Protected Function OnGet(ByVal key As Object, ByVal
22 currentValue As Object) As Object

23 [JScript] protected function OnGet(key : Object, currentValue : Object) : Object;

24 25 *Description*

Gets the element with the specified key and value in the **System.Collections.DictionaryBase** instance.

Return Value: An **System.Object** containing the element with the specified key and value.

The default implementation of this method returns *currentValue* . It is intended to be overridden by a derived class to perform additional action when the specified element is retrieved. The key of the element to get. The current value of the element associated with *key*.

OnInsert

[C#] protected virtual void OnInsert(object key, object value);

[C++] protected: virtual void OnInsert(Object* key, Object* value);

[VB] Overridable Protected Sub OnInsert(ByVal key As Object, ByVal value As Object)

[JScript] protected function OnInsert(key : Object, value : Object);

Description

Performs additional custom processes before inserting a new element into the **System.Collections.DictionaryBase** instance.

The default implementation of this method is intended to be overridden by a derived class to perform some action before the specified element is inserted.

The key of the element to insert. The value of the element to insert.

OnInsertComplete

[C#] protected virtual void OnInsertComplete(object key, object value);

1 [C++] protected: virtual void OnInsertComplete(Object* key, Object* value);

2 [VB] Overridable Protected Sub OnInsertComplete(ByVal key As Object, ByVal
3 value As Object)

4 [JScript] protected function OnInsertComplete(key : Object, value : Object);

5
6 *Description*

7 Performs additional custom processes after inserting a new element into the
8 **System.Collections.DictionaryBase** instance.

9 The default implementation of this method is intended to be overridden by
10 a derived class to perform some action after the specified element is inserted. The
11 key of the element to insert. The value of the element to insert.

12 **OnRemove**

13
14 [C#] protected virtual void OnRemove(object key, object value);

15 [C++] protected: virtual void OnRemove(Object* key, Object* value);

16 [VB] Overridable Protected Sub OnRemove(ByVal key As Object, ByVal value
17 As Object)

18 [JScript] protected function OnRemove(key : Object, value : Object);

19
20 *Description*

21 Performs additional custom processes before removing an element from the
22 **System.Collections.DictionaryBase** instance.

23 The default implementation of this method is intended to be overridden by
24 a derived class to perform some action before the specified element is removed.

25 The key of the element to remove. The value of the element to remove.

OnRemoveComplete

[C#] protected virtual void OnRemoveComplete(object key, object value);

[C++] protected: virtual void OnRemoveComplete(Object* key, Object* value);

[VB] Overridable Protected Sub OnRemoveComplete(ByVal key As Object,
ByVal value As Object)

[JScript] protected function OnRemoveComplete(key : Object, value : Object);

Description

Performs additional custom processes after removing an element from the **System.Collections.DictionaryBase** instance.

The default implementation of this method is intended to be overridden by a derived class to perform some action after the specified element is removed. The key of the element to remove. The value of the element to remove.

OnSet

[C#] protected virtual void OnSet(object key, object oldValue, object newValue);

[C++] protected: virtual void OnSet(Object* key, Object* oldValue, Object*
newValue);

[VB] Overridable Protected Sub OnSet(ByVal key As Object, ByVal oldValue As
Object, ByVal newValue As Object)

[JScript] protected function OnSet(key : Object, oldValue : Object, newValue :
Object);

Description

1 Performs additional custom processes before setting a value in the
2 **System.Collections.DictionaryBase** instance.

3 The default implementation of this method is intended to be overridden by
4 a derived class to perform some action before the specified element is set. The key
5 of the element to locate. The old value of the element associated with *key*. The
6 new value of the element associated with *key*.

7 OnSetComplete

8
9 [C#] protected virtual void OnSetComplete(object key, object oldValue, object
10 newValue);

11 [C++] protected: virtual void OnSetComplete(Object* key, Object* oldValue,
12 Object* newValue);

13 [VB] Overridable Protected Sub OnSetComplete(ByVal key As Object, ByVal
14 oldValue As Object, ByVal newValue As Object)

15 [JScript] protected function OnSetComplete(key : Object, oldValue : Object,
16 newValue : Object);

17 18 *Description*

19 Performs additional custom processes after setting a value in the
20 **System.Collections.DictionaryBase** instance.

21 The default implementation of this method is intended to be overridden by
22 a derived class to perform some action after the specified element is set. The key
23 of the element to locate. The old value of the element associated with *key*. The
24 new value of the element associated with *key*.

25 OnValidate

1
2 [C#] protected virtual void OnValidate(object key, object value);

3 [C++] protected: virtual void OnValidate(Object* key, Object* value);

4 [VB] Overridable Protected Sub OnValidate(ByVal key As Object, ByVal value
5 As Object)

6 [JScript] protected function OnValidate(key : Object, value : Object);

7
8 *Description*

9 Performs additional custom processes when validating the element with the
10 specified key and value.

11 The default implementation of this method is intended to be overridden by
12 a derived class to perform some action when the specified element is validated.

13 The key of the element to validate. The value of the element to validate.

14 IDictionary.Add

15
16 [C#] void IDictionary.Add(object key, object value);

17 [C++] void IDictionary::Add(Object* key, Object* value);

18 [VB] Sub Add(ByVal key As Object, ByVal value As Object) Implements
19 IDictionary.Add

20 [JScript] function IDictionary.Add(key : Object, value : Object);

21 IDictionary.Contains

22
23 [C#] bool IDictionary.Contains(object key);

24 [C++] bool IDictionary::Contains(Object* key);

25 [VB] Function Contains(ByVal key As Object) As Boolean Implements

IDictionary.Contains

[JScript] function IDictionary.Contains(key : Object) : Boolean;

IDictionary.Remove

[C#] void IDictionary.Remove(object key);

[C++] void IDictionary::Remove(Object* key);

[VB] Sub Remove(ByVal key As Object) Implements IDictionary.Remove

[JScript] function IDictionary.Remove(key : Object);

IEnumerable.GetEnumerator

[C#] IEnumerator IEnumerable.GetEnumerator();

[C++] IEnumerator* IEnumerable::GetEnumerator();

[VB] Function GetEnumerator() As IEnumerator Implements

IEnumerable.GetEnumerator

[JScript] function IEnumerable.GetEnumerator() : IEnumerator;

DictionaryEntry structure (System.Collections)

ToString

Description

Defines a dictionary key-and-value pair that can be set or retrieved.

The **System.Collections.IDictionaryEnumerator.Entry** method returns an instance of this class.

DictionaryEntry

Example Syntax:

ToString

[C#] public DictionaryEntry(object key, object value);

[C++] public: DictionaryEntry(Object* key, Object* value);

[VB] Public Sub New(ByVal key As Object, ByVal value As Object)

[JScript] public function DictionaryEntry(key : Object, value : Object);

Description

Initializes an instance of the **System.Collections.DictionaryEntry** class with the specified key and value. The object defined in each key-and-value pair. The definition associated with *key*.

Key

ToString

[C#] public object Key {get; set;}

[C++] public: __property Object* get_Key();public: __property void set_Key(Object*);

[VB] Public Property Key As Object

[JScript] public function get Key() : Object;public function set Key(Object);

Description

Gets or sets the key in the key-and-value pair.

Value

ToString

1
2 [C#] public object Value {get; set;}

3 [C++] public: __property Object* get_Value();public: __property void

4 set_Value(Object*);

5 [VB] Public Property Value As Object

6 [JScript] public function get Value() : Object;public function set Value(Object);

7
8 *Description*

9 Gets or sets the value in the key-and-value pair.

10 Hashtable class (System.Collections)

11 ToString

12
13
14 *Description*

15 Represents a collection of key-and-value pairs that are organized based on
16 the hash code of the key.

17 Each element is a key-and-value pair stored in a
18 **System.Collections.DictionaryEntry** object.

19 Hashtable

20 *Example Syntax:*

21 ToString

22
23 [C#] public Hashtable();

24 [C++] public: Hashtable();

25 [VB] Public Sub New()

[JScript] public function Hashtable(); Initializes a new instance of the **System.Collections.Hashtable** class.

Description

Creates an empty **System.Collections.Hashtable** with the default initial capacity and using the default load factor, the default hash code provider and the default comparer.

A hashtable's capacity is used to calculate the optimal number of hashtable buckets based on the load factor. The default initial capacity is zero. Capacity is automatically increased as required.

Hashtable

Example Syntax:

ToString

[C#] public Hashtable(IDictionary d);

[C++] public: Hashtable(IDictionary* d);

[VB] Public Sub New(ByVal d As IDictionary)

[JScript] public function Hashtable(d : IDictionary);

Description

Copies the elements from the specified dictionary to a new **System.Collections.Hashtable** with the same initial capacity as the number of elements copied and using the default load factor, the default hash code provider and the default comparer.

The initial capacity is set to the number of elements in the source dictionary. Capacity is automatically increased as required based on the load factor. The **System.Collections.IDictionary** to copy to a new

System.Collections.Hashtable.

Hashtable

Example Syntax:

ToString

[C#] public Hashtable(int capacity);

[C++] public: Hashtable(int capacity);

[VB] Public Sub New(ByVal capacity As Integer)

[JScript] public function Hashtable(capacity : int);

Description

Creates an empty **System.Collections.Hashtable** with the specified initial capacity and using the default load factor, the default hash code provider and the default comparer.

Specifying the initial capacity eliminates the need to perform a number of resizing operations while adding elements to the **System.Collections.Hashtable**. Capacity is automatically increased as required based on the load factor. The approximate number of elements that the **System.Collections.Hashtable** can initially contain.

Hashtable

Example Syntax:

ToString

1
2 [C#] public Hashtable(IDictionary d, float loadFactor);

3 [C++] public: Hashtable(IDictionary* d, float loadFactor);

4 [VB] Public Sub New(ByVal d As IDictionary, ByVal loadFactor As Single)

5 [JScript] public function Hashtable(d : IDictionary, loadFactor : float);

6
7 *Description*

8 Copies the elements from the specified dictionary to a new
9 **System.Collections.Hashtable** with the same initial capacity as the number of
10 elements copied and using the specified load factor, the default hash code provider
11 and the default comparer.

12 The initial capacity is set to the number of elements in the source
13 dictionary. Capacity is automatically increased as required based on the load
14 factor. The **System.Collections.IDictionary** to copy to a new
15 **System.Collections.Hashtable**. A number in the range from 0.1 through 1.0
16 indicating the maximum ratio of elements to buckets.

17 Hashtable

18 *Example Syntax:*

19 ToString

20
21 [C#] public Hashtable(IHashCodeProvider hcp, IComparer comparer);

22 [C++] public: Hashtable(IHashCodeProvider* hcp, IComparer* comparer);

23 [VB] Public Sub New(ByVal hcp As IHashCodeProvider, ByVal comparer As
24 IComparer)

25 [JScript] public function Hashtable(hcp : IHashCodeProvider, comparer :

1 IComparer);

2
3 *Description*

4 Creates an empty **System.Collections.Hashtable** with the default initial
5 capacity and using the default load factor, the specified hash code provider and the
6 specified comparer.

7 A hashtable's capacity is used to calculate the optimal number of hashtable
8 buckets based on the load factor. The default initial capacity is zero. Capacity is
9 automatically increased as required. The
10 **System.Collections.IHashCodeProvider** that supplies the hash codes for all keys
11 in the **System.Collections.Hashtable**. The **System.Collections.IComparer** to use
12 to determine whether two keys are equal.

13 Hashtable

14 *Example Syntax:*

15 ToString

16
17 [C#] public Hashtable(int capacity, float loadFactor);

18 [C++] public: Hashtable(int capacity, float loadFactor);

19 [VB] Public Sub New(ByVal capacity As Integer, ByVal loadFactor As Single)

20 [JScript] public function Hashtable(capacity : int, loadFactor : float);

21
22 *Description*

23 Creates an empty **System.Collections.Hashtable** with the specified initial
24 capacity and using the specified load factor, the default hash code provider and the
25 default comparer.

Specifying the initial capacity eliminates the need to perform a number of resizing operations while adding elements to the **System.Collections.Hashtable** . Capacity is automatically increased as required based on the load factor. The approximate number of elements that the **System.Collections.Hashtable** can initially contain. A number in the range from 0.1 through 1.0 indicating the maximum ratio of elements to buckets.

Hashtable

Example Syntax:

ToString

[C#] protected Hashtable(SerializationInfo info, StreamingContext context);

[C++] protected: Hashtable(SerializationInfo* info, StreamingContext context);

[VB] Protected Sub New(ByVal info As SerializationInfo, ByVal context As StreamingContext)

[JScript] protected function Hashtable(info : SerializationInfo, context : StreamingContext);

Hashtable

Example Syntax:

ToString

[C#] public Hashtable(IDictionary d, IHashCodeProvider hcp, IComparer comparer);

[C++] public: Hashtable(IDictionary* d, IHashCodeProvider* hcp, IComparer* comparer);

[VB] Public Sub New(ByVal d As IDictionary, ByVal hcp As

1 IHashCodeProvider, ByVal comparer As IComparer)

2 [JScript] public function Hashtable(d : IDictionary, hcp : IHashCodeProvider,
3 comparer : IComparer);

4
5 *Description*

6 Copies the elements from the specified dictionary to a new
7 **System.Collections.Hashtable** with the same initial capacity as the number of
8 elements copied and using the default load factor, the specified hash code provider
9 and the specified comparer.

10 The initial capacity is set to the number of elements in the source
11 dictionary. Capacity is automatically increased as required based on the load
12 factor. The **System.Collections.IDictionary** to copy to a new
13 **System.Collections.Hashtable**. The **System.Collections.IHashCodeProvider**
14 that supplies the hash codes for all keys in the **System.Collections.Hashtable**.
15 The **System.Collections.IComparer** to use to determine whether two keys are
16 equal.

17 Hashtable

18 *Example Syntax:*

19 ToString

20
21 [C#] public Hashtable(int capacity, IHashCodeProvider hcp, IComparer
22 comparer);

23 [C++] public: Hashtable(int capacity, IHashCodeProvider* hcp, IComparer*
24 comparer);

25 [VB] Public Sub New(ByVal capacity As Integer, ByVal hcp As

1 IHashCodeProvider, ByVal comparer As IComparer)

2 [JScript] public function Hashtable(capacity : int, hcp : IHashCodeProvider,
3 comparer : IComparer);

5 *Description*

6 Creates an empty **System.Collections.Hashtable** with the specified initial
7 capacity and using the default load factor, the specified hash code provider and the
8 specified comparer.

9 Specifying the initial capacity eliminates the need to perform a number of
10 resizing operations while adding elements to the **System.Collections.Hashtable** .
11 Capacity is automatically increased as required based on the load factor. The
12 approximate number of elements that the **System.Collections.Hashtable** can
13 initially contain. The **System.Collections.IHashCodeProvider** that supplies the
14 hash codes for all keys in the **System.Collections.Hashtable**. The
15 **System.Collections.IComparer** to use to determine whether two keys are equal.

16 Hashtable

17 *Example Syntax:*

18 ToString

19
20 [C#] public Hashtable(IDictionary d, float loadFactor, IHashCodeProvider hcp,
21 IComparer comparer);

22 [C++] public: Hashtable(IDictionary* d, float loadFactor, IHashCodeProvider*
23 hcp, IComparer* comparer);

24 [VB] Public Sub New(ByVal d As IDictionary, ByVal loadFactor As Single,
25 ByVal hcp As IHashCodeProvider, ByVal comparer As IComparer)

1 [JScript] public function Hashtable(d : IDictionary, loadFactor : float, hcp :
2 IHashCodeProvider, comparer : IComparer);

4 *Description*

5 Copies the elements from the specified dictionary to a new
6 **System.Collections.Hashtable** with the same initial capacity as the number of
7 elements copied and using the specified load factor, the specified hash code
8 provider and the specified comparer.

9 The initial capacity is set to the number of elements in the source
10 dictionary. Capacity is automatically increased as required based on the load
11 factor. The **System.Collections.IDictionary** to copy to a new
12 **System.Collections.Hashtable**. A number in the range from 0.1 through 1.0
13 indicating the maximum ratio of elements to buckets. The
14 **System.Collections.IHashCodeProvider** that supplies the hash codes for all keys
15 in the **System.Collections.Hashtable**. The **System.Collections.IComparer** to use
16 to determine whether two keys are equal.

17 Hashtable

18 *Example Syntax:*

19 ToString

20
21 [C#] public Hashtable(int capacity, float loadFactor, IHashCodeProvider hcp,
22 IComparer comparer);
23 [C++] public: Hashtable(int capacity, float loadFactor, IHashCodeProvider* hcp,
24 IComparer* comparer);
25 [VB] Public Sub New(ByVal capacity As Integer, ByVal loadFactor As Single,

ByVal hcp As IHashCodeProvider, ByVal comparer As IComparer)
[JScript] public function Hashtable(capacity : int, loadFactor : float, hcp :
IHashCodeProvider, comparer : IComparer);

Description

Creates an empty **System.Collections.Hashtable** with the specified initial capacity and using the specified load factor, the specified hash code provider and the specified comparer.

Specifying the initial capacity eliminates the need to perform a number of resizing operations while adding elements to the **System.Collections.Hashtable**. Capacity is automatically increased as required based on the load factor. The approximate number of elements that the **System.Collections.Hashtable** can initially contain. A number in the range from 0.1 through 1.0 indicating the maximum ratio of elements to buckets. The **System.Collections.IHashCodeProvider** that supplies the hash codes for all keys in the **System.Collections.Hashtable**. The **System.Collections.IComparer** to use to determine whether two keys are equal.

comparer

ToString

[C#] protected IComparer comparer {get; set;}

[C++] protected: __property IComparer* get_comparer();protected: __property
void set_comparer(IComparer*);

[VB] Protected Property comparer As IComparer

[JScript] protected function get comparer() : IComparer;protected function set

1 comparer(IComparer);

3 *Description*

4 Gets or sets the comparer to use for the **System.Collections.Hashtable** .

5 Count

6 ToString

8 [C#] public virtual int Count {get;}

9 [C++] public: __property virtual int get_Count();

10 [VB] Overridable Public ReadOnly Property Count As Integer

11 [JScript] public function get Count() : int;

13 *Description*

14 Gets the number of key-and-value pairs contained in the
15 **System.Collections.Hashtable** .

16 hcp

17 ToString

19 [C#] protected IHashCodeProvider hcp {get; set;}

20 [C++] protected: __property IHashCodeProvider* get_hcp();protected: __property

21 void set_hcp(IHashCodeProvider*);

22 [VB] Protected Property hcp As IHashCodeProvider

23 [JScript] protected function get hcp() : IHashCodeProvider;protected function set

24 hcp(IHashCodeProvider);

1
2 *Description*

3 Gets or sets the object that can dispense hash codes.

4 IsFixedSize

5 ToString

6
7 [C#] public virtual bool IsFixedSize {get;}

8 [C++] public: __property virtual bool get_IsFixedSize();

9 [VB] Overridable Public ReadOnly Property IsFixedSize As Boolean

10 [JScript] public function get IsFixedSize() : Boolean;

11
12 *Description*

13 Gets a value indicating whether the **System.Collections.Hashtable** has a
14 fixed size.

15 A collection with a fixed size does not allow the addition or removal of
16 elements, but it allows the modification of existing elements.

17 IsReadOnly

18 ToString

19
20 [C#] public virtual bool IsReadOnly {get;}

21 [C++] public: __property virtual bool get_IsReadOnly();

22 [VB] Overridable Public ReadOnly Property IsReadOnly As Boolean

23 [JScript] public function get IsReadOnly() : Boolean;

24
25 *Description*

1 Gets a value indicating whether the **System.Collections.Hashtable** is read-
2 only.

3 IsSynchronized

4 ToString

5
6 [C#] public virtual bool IsSynchronized {get;}

7 [C++] public: __property virtual bool get_IsSynchronized();

8 [VB] Overridable Public ReadOnly Property IsSynchronized As Boolean

9 [JScript] public function get IsSynchronized() : Boolean;

10
11 *Description*

12 Gets a value indicating whether access to the
13 **System.Collections.Hashtable** is synchronized (thread-safe).

14 A **System.Collections.Hashtable** can safely support one writer and
15 multiple readers concurrently. To support multiple writers, all operations must be
16 done through the wrapper returned by the
17 **System.Collections.Hashtable.Synchronized(System.Collections.Hashtable)**
18 method.

19 Item

20 ToString

21
22 [C#] public virtual object this[object key] {get; set;}

23 [C++] public: __property virtual Object* get_Item(Object* key);public:

24 __property virtual void set_Item(Object* key, Object*);

25 [VB] Overridable Public Default Property Item(ByVal key As Object) As Object

1 [JScript] returnValue = HashtableObject.Item(key);HashtableObject.Item(key) =
2 returnValue;

4 *Description*

5 Gets or sets the value associated with the specified key.

6 This property provides the ability to access a specific element in the
7 collection by using the following syntax: myCollection[key] . The key whose
8 value to get or set.

9 Keys

10 ToString

11
12 [C#] public virtual ICollection Keys {get;}

13 [C++] public: __property virtual ICollection* get_Keys();

14 [VB] Overridable Public ReadOnly Property Keys As ICollection

15 [JScript] public function get Keys() : ICollection;

16 17 *Description*

18 Gets an **System.Collections.ICollection** containing the keys in the
19 **System.Collections.Hashtable** .

20 The order of the keys in the **System.Collections.ICollection** is unspecified,
21 but it is the same order as the associated values in the
22 **System.Collections.ICollection** returned by the
23 **System.Collections.Hashtable.Values** method.

24 SyncRoot

25 ToString

1
2 [C#] public virtual object SyncRoot {get;}

3 [C++] public: __property virtual Object* get_SyncRoot();

4 [VB] Overridable Public ReadOnly Property SyncRoot As Object

5 [JScript] public function get SyncRoot() : Object;

6
7 *Description*

8 Gets an object that can be used to synchronize access to the

9 **System.Collections.Hashtable** .

10 To create a synchronized version of the **System.Collections.Hashtable** ,
11 use the

12 **System.Collections.Hashtable.Synchronized(System.Collections.Hashtable)**

13 method. However, derived classes can provide their own synchronized version of
14 the **System.Collections.Hashtable** using the

15 **System.Collections.Hashtable.SyncRoot** property. The synchronizing code must
16 perform operations on the **System.Collections.Hashtable.SyncRoot** of the

17 **System.Collections.Hashtable** , not directly on the

18 **System.Collections.Hashtable** . This ensures proper operation of collections that

19 are derived from other objects. Specifically, it maintains proper synchronization
20 with other threads that might be simultaneously modifying the

21 **System.Collections.Hashtable** object.

22 Values

23 ToString

24
25 [C#] public virtual ICollection Values {get;}

1 [C++] public: __property virtual ICollection* get_Values();

2 [VB] Overridable Public ReadOnly Property Values As ICollection

3 [JScript] public function get Values() : ICollection;

4
5 *Description*

6 Gets an **System.Collections.ICollection** containing the values in the
7 **System.Collections.Hashtable** .

8 The order of the values in the **System.Collections.ICollection** is
9 unspecified, but it is the same order as the associated keys in the
10 **System.Collections.ICollection** returned by the
11 **System.Collections.Hashtable.Keys** method.

12 **Add**

13
14 [C#] public virtual void Add(object key, object value);

15 [C++] public: virtual void Add(Object* key, Object* value);

16 [VB] Overridable Public Sub Add(ByVal key As Object, ByVal value As Object)

17 [JScript] public function Add(key : Object, value : Object);

18
19 *Description*

20 Adds an element with the specified key and value into the
21 **System.Collections.Hashtable** .

22 An object that has no correlation between its state and its hash code value
23 should typically not be used as the key. For example, String objects are better than
24 StringBuilder objects for use as keys. The key of the element to add. The value of
25 the element to add.

Clear

[C#] public virtual void Clear();

[C++] public: virtual void Clear();

[VB] Overridable Public Sub Clear()

[JScript] public function Clear();

Description

Removes all elements from the **System.Collections.Hashtable** .

System.Collections.Hashtable.Count is set to zero.

Clone

[C#] public virtual object Clone();

[C++] public: virtual Object* Clone();

[VB] Overridable Public Function Clone() As Object

[JScript] public function Clone() : Object;

Description

Creates a shallow copy of the **System.Collections.Hashtable** .

Return Value: A shallow copy of the **System.Collections.Hashtable** .

A shallow copy of a collection is a new collection containing references to the same elements as the original collection. The elements themselves or anything referenced by the elements are not copied. In contrast, a deep copy of a collection copies the elements and everything directly or indirectly referenced by the elements.

Contains

[C#] public virtual bool Contains(object key);

[C++] public: virtual bool Contains(Object* key);

[VB] Overridable Public Function Contains(ByVal key As Object) As Boolean

[JScript] public function Contains(key : Object) : Boolean;

Description

Determines whether the **System.Collections.Hashtable** contains a specific key.

Return Value: **true** if the **System.Collections.Hashtable** contains an element with the specified key; otherwise, **false**.

This implementation is close to O(1) in most cases. The key to locate in the **System.Collections.Hashtable**.

ContainsKey

[C#] public virtual bool ContainsKey(object key);

[C++] public: virtual bool ContainsKey(Object* key);

[VB] Overridable Public Function ContainsKey(ByVal key As Object) As

Boolean

[JScript] public function ContainsKey(key : Object) : Boolean;

Description

Determines whether the **System.Collections.Hashtable** contains a specific key.

Return Value: **true** if the **System.Collections.Hashtable** contains an element with the specified key; otherwise, **false** .

This implementation is close to $O(1)$ in most cases. The key to locate in the **System.Collections.Hashtable**.

ContainsValue

[C#] public virtual bool ContainsValue(object value);

[C++] public: virtual bool ContainsValue(Object* value);

[VB] Overridable Public Function ContainsValue(ByVal value As Object) As

Boolean

[JScript] public function ContainsValue(value : Object) : Boolean;

Description

Determines whether the **System.Collections.Hashtable** contains a specific value.

Return Value: **true** if the **System.Collections.Hashtable** contains an element with the specified *value* ; otherwise, **false** .

This method performs a linear search; therefore, the average execution time is proportional to **System.Collections.Hashtable.Count** . That is, this method is an $O(n)$ operation, where n is **System.Collections.Hashtable.Count** . The value to locate in the **System.Collections.Hashtable**.

CopyTo

[C#] public virtual void CopyTo(Array array, int arrayIndex);

[C++] public: virtual void CopyTo(Array* array, int arrayIndex);

1 [VB] Overridable Public Sub CopyTo(ByVal array As Array, ByVal arrayIndex
2 As Integer)

3 [JScript] public function CopyTo(array : Array, arrayIndex : int);

4
5 *Description*

6 Copies the **System.Collections.Hashtable** elements to a one-dimensional
7 **System.Array** instance at the specified index.

8 The elements are copied to the **System.Array** in the same order in which
9 the enumerator iterates through the **System.Collections.Hashtable** . The one-
10 dimensional **System.Array** that is the destination of the
11 **System.Collections.DictionaryEntry** objects copied from
12 **System.Collections.Hashtable**. The **System.Array** must have zero-based
13 indexing. The zero-based index in *array* at which copying begins.

14 **GetEnumerator**

15
16 [C#] public virtual IDictionaryEnumerator GetEnumerator();

17 [C++] public: virtual IDictionaryEnumerator* GetEnumerator();

18 [VB] Overridable Public Function GetEnumerator() As IDictionaryEnumerator

19 [JScript] public function GetEnumerator() : IDictionaryEnumerator;

20
21 *Description*

22 Returns an enumerator that can iterate through the
23 **System.Collections.Hashtable** .

24 *Return Value:* An **System.Collections.IDictionaryEnumerator** for the
25 **System.Collections.Hashtable** .

Enumerators are intended to be used only to read data in the collection.

Enumerators cannot be used to modify the underlying collection.

GetHash

[C#] protected virtual int GetHash(object key);

[C++] protected: virtual int GetHash(Object* key);

[VB] Overridable Protected Function GetHash(ByVal key As Object) As Integer

[JScript] protected function GetHash(key : Object) : int;

Description

Returns the hash code for the specified key.

Return Value: The hash code for *key* .

If the hashtable was created with a specific

System.Collections.IHashCodeProvider implementation, this method uses that hash code provider; otherwise, it uses the **System.Object.GetHashCode** implementation of *key* . The **System.Object** for which a hash code is to be returned.

GetObjectData

[C#] public virtual void GetObjectData(SerializationInfo info, StreamingContext context);

[C++] public: virtual void GetObjectData(SerializationInfo* info, StreamingContext context);

[VB] Overridable Public Sub GetObjectData(ByVal info As SerializationInfo, ByVal context As StreamingContext)

1 [JScript] public function GetObjectData(info : SerializationInfo, context :
2 StreamingContext);

4 *Description*

5 Implements the **System.Runtime.Serialization.ISerializable** interface and
6 returns the data needed to serialize the **System.Collections.Hashtable** . A
7 **System.Runtime.Serialization.SerializationInfo** object containing the
8 information required to serialize the **System.Collections.Hashtable**. A
9 **System.Runtime.Serialization.StreamingContext** object containing the source
10 and destination of the serialized stream associated with the
11 **System.Collections.Hashtable**.

12 **KeyEquals**

13
14 [C#] protected virtual bool KeyEquals(object item, object key);

15 [C++] protected: virtual bool KeyEquals(Object* item, Object* key);

16 [VB] Overridable Protected Function KeyEquals(ByVal item As Object, ByVal
17 key As Object) As Boolean

18 [JScript] protected function KeyEquals(item : Object, key : Object) : Boolean;

19 20 *Description*

21 Compares a specific **System.Object** with a specific key in the
22 **System.Collections.Hashtable** .

23 *Return Value:* **true** if *item* and *key* are equal; otherwise, **false** .

24 If the hashtable was created with a specific
25 **System.Collections.IComparer** implementation, this method uses that comparer;

that is,

System.Collections.IComparer.Compare(System.Object, System.Object) (*item*, *key*). Otherwise, it uses *item.Equals(key)* . The **System.Object** to compare with *key*. The key in the **System.Collections.Hashtable** to compare with *item*.

OnDeserialization

[C#] public virtual void OnDeserialization(object sender);

[C++] public: virtual void OnDeserialization(Object* sender);

[VB] Overridable Public Sub OnDeserialization(ByVal sender As Object)

[JScript] public function OnDeserialization(sender : Object);

Description

Implements the **System.Runtime.Serialization.ISerializable** interface and raises the deserialization event when the deserialization is complete. The source of the deserialization event.

Remove

[C#] public virtual void Remove(object key);

[C++] public: virtual void Remove(Object* key);

[VB] Overridable Public Sub Remove(ByVal key As Object)

[JScript] public function Remove(key : Object);

Description

Removes the element with the specified key from the **System.Collections.Hashtable** .

If the **System.Collections.Hashtable** does not contain an element with the specified key, the **System.Collections.Hashtable** remains unchanged. No exception is thrown. The key of the element to remove.

Synchronized

[C#] public static Hashtable Synchronized(Hashtable table);

[C++] public: static Hashtable* Synchronized(Hashtable* table);

[VB] Public Shared Function Synchronized(ByVal table As Hashtable) As Hashtable

[JScript] public static function Synchronized(table : Hashtable) : Hashtable;

Description

Returns a synchronized (thread-safe) wrapper for the **System.Collections.Hashtable**.

Return Value: A synchronized (thread-safe) wrapper for the **System.Collections.Hashtable**.

A **System.Collections.Hashtable** can safely support one writer and multiple readers concurrently. To support multiple writers, all operations must be done through this wrapper only. The **System.Collections.Hashtable** to synchronize.

IEnumerable.GetEnumerator

[C#] IEnumerator IEnumerable.GetEnumerator();

[C++] IEnumerator* IEnumerable::GetEnumerator();

[VB] Function GetEnumerator() As IEnumerator Implements

1 IEnumerable.GetEnumerator

2 [JScript] function IEnumerable.GetEnumerator() : IEnumerator;

3 ICollection interface (System.Collections)

4 ToString

7 *Description*

8 Defines size, enumerators and synchronization methods for all collections.

9 The **System.Collections.ICollection** interface is the base interface for
10 classes in the **System.Collections** namespace.

11 Count

12 ToString

14 [C#] int Count {get;}

15 [C++] int get_Count();

16 [VB] ReadOnly Property Count As Integer

17 [JScript] abstract function get Count() : int;

19 *Description*

20 When implemented by a class, gets the number of elements contained in the
21 **System.Collections.ICollection** .

22 IsSynchronized

23 ToString

25 [C#] bool IsSynchronized {get;}

1 [C++] bool get_IsSynchronized();

2 [VB] ReadOnly Property IsSynchronized As Boolean

3 [JScript] abstract function get IsSynchronized() : Boolean;

4
5 *Description*

6 When implemented by a class, gets a value indicating whether access to the
7 **System.Collections.ICollection** is synchronized (thread-safe).

8 **System.Collections.ICollection.SyncRoot** returns an object, which can be
9 used to synchronize access to the **System.Collections.ICollection** .

10 SyncRoot

11 ToString

12
13 [C#] object SyncRoot {get;}

14 [C++] Object* get_SyncRoot();

15 [VB] ReadOnly Property SyncRoot As Object

16 [JScript] abstract function get SyncRoot() : Object;

17
18 *Description*

19 When implemented by a class, gets an object that can be used to
20 synchronize access to the **System.Collections.ICollection** .

21 For collections whose underlying store is not publicly available, the
22 expected implementation is to return the current instance. Note that the pointer to
23 the current instance might not be sufficient for collections that wrap other
24 collections; those should return the underlying collection's **SyncRoot** property.

25 CopyTo

```

1
2 [C#] void CopyTo(Array array, int index);
3 [C++] void CopyTo(Array* array, int index);
4 [VB] Sub CopyTo(ByVal array As Array, ByVal index As Integer)
5 [JScript] function CopyTo(array : Array, index : int);
6

```

Description

When implemented by a class, copies the elements of the **System.Collections.ICollection** to an **System.Array**, starting at a particular **System.Array** index. The one-dimensional **System.Array** that is the destination of the elements copied from **System.Collections.ICollection**. The **System.Array** must have zero-based indexing. The zero-based index in *array* at which copying begins.

IEnumerator interface (System.Collections)

CopyTo

Description

Exposes a method that compares two objects.

This interface is used in conjunction with the **System.Array.Sort(System.Array)** and **System.Array.BinarySearch(System.Array, System.Object)** methods. It provides a way to customize the sort order of a collection.

Compare

1
2 [C#] int Compare(object x, object y);

3 [C++] int Compare(Object* x, Object* y);

4 [VB] Function Compare(ByVal x As Object, ByVal y As Object) As Integer

5 [JScript] function Compare(x : Object, y : Object) : int;

6
7 *Description*

8 Compares two objects and returns a value indicating whether one is less
9 than, equal to or greater than the other.

10 *Return Value:* Value Condition Less than zero *x* is less than *y* .

11 The preferred implementation is to use the
12 **System.IComparable.CompareTo(System.Object)** method of one of the
13 parameters. First object to compare. Second object to compare.

14 IDictionary interface (System.Collections)

15 Compare

16
17
18 *Description*

19 Represents a collection of key-and-value pairs.

20 The **System.Collections.IDictionary** class is the base interface for
21 collections of key-and-value pairs.

22 IsFixedSize

23 Compare

24
25 [C#] bool IsFixedSize {get;}

1 [C++] bool get_IsFixedSize();

2 [VB] ReadOnly Property IsFixedSize As Boolean

3 [JScript] abstract function get IsFixedSize() : Boolean;

4
5 *Description*

6 When implemented by a class, gets a value indicating whether the

7 **System.Collections.IDictionary** has a fixed size.

8 A collection with a fixed size does not allow the addition or removal of
9 elements, but it allows the modification of existing elements.

10 IsReadOnly

11 Compare

12
13 [C#] bool IsReadOnly {get;}

14 [C++] bool get_IsReadOnly();

15 [VB] ReadOnly Property IsReadOnly As Boolean

16 [JScript] abstract function get IsReadOnly() : Boolean;

17
18 *Description*

19 When implemented by a class, gets a value indicating whether the

20 **System.Collections.IDictionary** is read-only.

21 Item

22 Compare

23
24 [C#] object this[object key] {get; set;}

25 [C++] Object* get_Item(Object* key);void set_Item(Object* key, Object*);

1 [VB] Default Property Item(ByVal key As Object) As Object

2 [JScript] abstract returnValue =

3 IDictionaryObject.Item(key);IDictionaryObject.Item(key) = returnValue;

4
5 *Description*

6 When implemented by a class, gets or sets the element with the specified
7 key.

8 This property provides the ability to access a specific element in the
9 collection by using the following syntax: myCollection[key] . The key of the
10 element to get or set.

11 Keys

12 Compare

13
14 [C#] ICollection Keys {get;}

15 [C++] ICollection* get_Keys();

16 [VB] ReadOnly Property Keys As ICollection

17 [JScript] abstract function get Keys() : ICollection;

18
19 *Description*

20 When implemented by a class, gets an **System.Collections.ICollection**
21 containing the keys of the **System.Collections.IDictionary** .

22 The order of the keys in the returned **System.Collections.ICollection** is
23 unspecified, but it is guaranteed to be the same order as the corresponding values
24 in the **System.Collections.ICollection** returned by the
25 **System.Collections.IDictionary.Values** method.

Values

Compare

[C#] ICollection Values {get;}

[C++] ICollection* get_Values();

[VB] ReadOnly Property Values As ICollection

[JScript] abstract function get Values() : ICollection;

Description

When implemented by a class, gets an **System.Collections.ICollection** containing the values in the **System.Collections.IDictionary** .

The order of the values in the returned **System.Collections.ICollection** is unspecified, but it is guaranteed to be the same order as the corresponding keys in the **System.Collections.ICollection** returned by the **System.Collections.IDictionary.Keys** method.

Add

[C#] void Add(object key, object value);

[C++] void Add(Object* key, Object* value);

[VB] Sub Add(ByVal key As Object, ByVal value As Object)

[JScript] function Add(key : Object, value : Object);

Description

When implemented by a class, adds an element with the provided key and value to the **System.Collections.IDictionary** .

The **System.Collections.IDictionary.Item(System.Object)** property can also be used to add new elements by setting the value of a key that does not exist in the dictionary. For example: `myCollection["myNonexistentKey"] = myValue` . However, if the specified key already exists in the dictionary, setting the **System.Collections.IDictionary.Item(System.Object)** property overwrites the old value. In contrast, the **System.Collections.IDictionary.Add(System.Object,System.Object)** method does not modify existing elements. The **System.Object** to use as the key of the element to add. The **System.Object** to use as the value of the element to add.

Clear

[C#] void Clear();

[C++] void Clear();

[VB] Sub Clear()

[JScript] function Clear();

Description

When implemented by a class, removes all elements from the **System.Collections.IDictionary** .

Contains

[C#] bool Contains(object key);

[C++] bool Contains(Object* key);

[VB] Function Contains(ByVal key As Object) As Boolean

[JScript] function Contains(key : Object) : Boolean;

Description

When implemented by a class, determines whether the **System.Collections.IDictionary** contains an element with the specified key.

Return Value: **true** if the **System.Collections.IDictionary** contains an element with the key; otherwise, **false** . The key to locate in the

System.Collections.IDictionary.

GetEnumerator

[C#] IDictionaryEnumerator GetEnumerator();

[C++] IDictionaryEnumerator* GetEnumerator();

[VB] Function GetEnumerator() As IDictionaryEnumerator

[JScript] function GetEnumerator() : IDictionaryEnumerator;

Description

When implemented by a class, returns an **System.Collections.IDictionaryEnumerator** for the **System.Collections.IDictionary** .

Return Value: An **System.Collections.IDictionaryEnumerator** for the **System.Collections.IDictionary** .

Enumerators are intended to be used only to read data in the collection.

Enumerators cannot be used to modify the underlying collection.

Remove

[C#] void Remove(object key);

1 [C++] void Remove(Object* key);

2 [VB] Sub Remove(ByVal key As Object)

3 [JScript] function Remove(key : Object);

4
5 *Description*

6 When implemented by a class, removes the element with the specified key
7 from the **System.Collections.IDictionary** .

8 In collections such as lists, queues and stacks, the elements that follow the
9 removed element move up to occupy the vacated spot. The key of the element to
10 remove.

11 IDictionaryEnumerator interface (System.Collections)

12 Remove

13
14
15 *Description*

16 Enumerates the elements of a dictionary.

17 Enumerators are intended to be used only to read data in the collection.

18 Enumerators cannot be used to modify the underlying collection.

19 Entry

20 Remove

21
22 [C#] DictionaryEntry Entry {get;}

23 [C++] DictionaryEntry get_Entry();

24 [VB] ReadOnly Property Entry As DictionaryEntry

25 [JScript] abstract function get Entry() : DictionaryEntry;

1
2 *Description*

3 When implemented by a class, gets both the key and the value of the
4 current dictionary entry.

5 After an enumerator is created or after a
6 **System.Collections.IEnumerator.Reset** ,
7 **System.Collections.IEnumerator.MoveNext** must be called to advance the
8 enumerator to the first element of the collection before reading the value of
9 **System.Collections.IDictionaryEnumerator.Entry** ; otherwise,
10 **System.Collections.IDictionaryEnumerator.Entry** is undefined.

11 Key

12 Remove

13
14 [C#] object Key {get;}

15 [C++] Object* get_Key();

16 [VB] ReadOnly Property Key As Object

17 [JScript] abstract function get Key() : Object;

18
19 *Description*

20 When implemented by a class, gets the key of the current dictionary entry.

21 After an enumerator is created or after a

22 **System.Collections.IEnumerator.Reset** ,
23 **System.Collections.IEnumerator.MoveNext** must be called to advance the
24 enumerator to the first element of the collection before reading the value of
25

1 **System.Collections.IDictionaryEnumerator.Key** ; otherwise,

2 **System.Collections.IDictionaryEnumerator.Key** is undefined.

3 Value

4 Remove

6 [C#] object Value {get;}

7 [C++] Object* get_Value();

8 [VB] ReadOnly Property Value As Object

9 [JScript] abstract function get Value() : Object;

11 *Description*

12 When implemented by a class, gets the value of the current dictionary
13 entry.

14 After an enumerator is created or after a

15 **System.Collections.IEnumerator.Reset** ,

16 **System.Collections.IEnumerator.MoveNext** must be called to advance the
17 enumerator to the first element of the collection before reading the value of

18 **System.Collections.IDictionaryEnumerator.Value** ; otherwise,

19 **System.Collections.IDictionaryEnumerator.Value** is undefined.

20 IEnumerable interface (System.Collections)

21 Remove

24 *Description*

Exposes the enumerator, which supports a simple iteration over a collection.

System.Collections.IEnumerable must be implemented to support the **ForEach** semantics of Microsoft Visual Basic. COM classes that allow enumerators also implement this interface.

GetEnumerator

[C#] IEnumerator GetEnumerator();

[C++] IEnumerator* GetEnumerator();

[VB] Function GetEnumerator() As IEnumerator

[JScript] function GetEnumerator() : IEnumerator;

Description

Returns an enumerator that can iterate through a collection.

Return Value: An **System.Collections.IEnumerator** that can be used to iterate through the collection.

Enumerators are intended to be used only to read data in the collection.

Enumerators cannot be used to modify the underlying collection.

IEnumerator interface (System.Collections)

GetEnumerator

Description

Supports a simple iteration over a collection.

System.Collections.IEnumerator is the base interface for all enumerators.

Current

GetEnumerator

[C#] object Current {get;}

[C++] Object* get_Current();

[VB] ReadOnly Property Current As Object

[JScript] abstract function get Current() : Object;

Description

Gets the current element in the collection.

After an enumerator is created or after a

System.Collections.IEnumerator.Reset ,

System.Collections.IEnumerator.MoveNext must be called to advance the enumerator to the first element of the collection before reading the value of

System.Collections.IEnumerator.Current ; otherwise,

System.Collections.IEnumerator.Current is undefined.

MoveNext

[C#] bool MoveNext();

[C++] bool MoveNext();

[VB] Function MoveNext() As Boolean

[JScript] function MoveNext() : Boolean;

Description

Advances the enumerator to the next element of the collection.

Return Value: **true** if the enumerator was successfully advanced to the next element; **false** if the enumerator has passed the end of the collection.

After an enumerator is created or after a call to **System.Collections.IEnumerator.Reset**, an enumerator is positioned before the first element of the collection, and the first call to **System.Collections.IEnumerator.MoveNext** moves the enumerator over the first element of the collection.

Reset

[C#] void Reset();

[C++] void Reset();

[VB] Sub Reset()

[JScript] function Reset();

Description

Sets the enumerator to its initial position, which is before the first element in the collection.

All calls to **System.Collections.IEnumerator.Reset** must result in the same state for the enumerator. This may involve taking a new snapshot of the collection or moving to the beginning of the collection.

IHashCodeProvider interface (System.Collections)

Reset

Description

Supplies a hash code for an object, using a custom hash function.

The **System.Collections.IHashCodeProvider** interface is used in conjunction with **System.Collections.Hashtable**. The objects used as keys by a **System.Collections.Hashtable** must implement or inherit the **System.Object.GetHashCode** and **System.Object.Equals(System.Object)** methods. However, if the **System.Collections.Hashtable** constructor is passed a reference to an object that implements both the **System.Collections.IHashCodeProvider** interface and the **System.Collections.IComparer** interface, then **System.Collections.IHashCodeProvider.GetHashCode(System.Object)** and **System.Collections.IComparer.Compare(System.Object, System.Object)** can be used instead.

GetHashCode

[C#] int GetHashCode(object obj);

[C++] int GetHashCode(Object* obj);

[VB] Function GetHashCode(ByVal obj As Object) As Integer

[JScript] function GetHashCode(obj : Object) : int;

Description

Returns a hash code for the specified object.

Return Value: A hash code for the specified object.

The return value from this method must not be persisted for two reasons. First, the hash function of a class might be altered to generate a better distribution, rendering any values from the old hash function useless. Second, the default implementation of this class does not guarantee that the same value will be returned by different instances. The **System.Object** for which a hash code is to be returned.

ICollection interface (System.Collections)

GetHashCode

Description

Represents a collection of objects that can be individually accessed by index.

System.Collections.ICollection is a descendant of the **System.Collections.IEnumerable** interface and is the base interface of all lists.

IsFixedSize

GetHashCode

[C#] bool IsFixedSize {get;}

[C++] bool get_IsFixedSize();

[VB] ReadOnly Property IsFixedSize As Boolean

[JScript] abstract function get IsFixedSize() : Boolean;

Description

When implemented by a class, gets a value indicating whether the **System.Collections.ICollection** has a fixed size.

A collection with a fixed size does not allow the addition or removal of elements, but it allows the modification of existing elements.

IsReadOnly

GetHashCode

[C#] bool IsReadOnly {get;}

[C++] bool get_IsReadOnly();

[VB] ReadOnly Property IsReadOnly As Boolean

[JScript] abstract function get IsReadOnly() : Boolean;

Description

When implemented by a class, gets a value indicating whether the **System.Collections.ICollection** is read-only.

Item

GetHashCode

[C#] object this[int index] {get; set;}

[C++] Object* get_Item(int index);void set_Item(int index, Object*);

[VB] Default Property Item(ByVal index As Integer) As Object

[JScript] abstract returnValue = ICollection.Item(index);ICollection.Item(index) = returnValue;

Description

When implemented by a class, gets or sets the element at the specified index.

This property provides the ability to access a specific element in the collection by using the following syntax: myCollection[index] . The zero-based index of the element to get or set.

Add

[C#] int Add(object value);

[C++] int Add(Object* value);

[VB] Function Add(ByVal value As Object) As Integer

[JScript] function Add(value : Object) : int;

Description

When implemented by a class, adds an item to the **System.Collections.IList** .

Return Value: The position into which the new element was inserted. The **System.Object** to add to the **System.Collections.IList**.

Clear

[C#] void Clear();

[C++] void Clear();

[VB] Sub Clear()

[JScript] function Clear();

Description

When implemented by a class, removes all items from the
System.Collections.IList .

Contains

[C#] bool Contains(object value);

[C++] bool Contains(Object* value);

[VB] Function Contains(ByVal value As Object) As Boolean

[JScript] function Contains(value : Object) : Boolean;

Description

When implemented by a class, determines whether the
System.Collections.IList contains a specific value.

Return Value: **true** if the **System.Object** is found in the **System.Collections.IList**
; otherwise, **false** . The **System.Object** to locate in the **System.Collections.IList**.

IndexOf

[C#] int IndexOf(object value);

[C++] int IndexOf(Object* value);

[VB] Function IndexOf(ByVal value As Object) As Integer

[JScript] function IndexOf(value : Object) : int;

Description

When implemented by a class, determines the index of a specific item in
the **System.Collections.IList** .

1 *Return Value:* The index of *value* if found in the list; otherwise, -1. The

2 **System.Object** to locate in the **System.Collections.IList**.

3 Insert

4
5 [C#] void Insert(int index, object value);

6 [C++] void Insert(int index, Object* value);

7 [VB] Sub Insert(ByVal index As Integer, ByVal value As Object)

8 [JScript] function Insert(index : int, value : Object);

9
10 *Description*

11 When implemented by a class, inserts an item to the
12 **System.Collections.IList** at the specified position.

13 If *index* equals the number of items in the **System.Collections.IList** , then
14 *value* is appended to the end. The zero-based index at which *value* should be
15 inserted. The **System.Object** to insert into the **System.Collections.IList**.

16 Remove

17
18 [C#] void Remove(object value);

19 [C++] void Remove(Object* value);

20 [VB] Sub Remove(ByVal value As Object)

21 [JScript] function Remove(value : Object);

22
23 *Description*

24 When implemented by a class, removes the first occurrence of a specific
25 object from the **System.Collections.IList** .

1 In collections such as lists, queues and stacks, the elements that follow the
2 removed element move up to occupy the vacated spot. The **System.Object** to
3 remove from the **System.Collections.IList**.

4 RemoveAt

5
6 [C#] void RemoveAt(int index);

7 [C++] void RemoveAt(int index);

8 [VB] Sub RemoveAt(ByVal index As Integer)

9 [JScript] function RemoveAt(index : int);

10 11 *Description*

12 When implemented by a class, removes the **System.Collections.IList** item
13 at the specified index.

14 In collections such as lists, queues and stacks, the elements that follow the
15 removed element move up to occupy the vacated spot. The zero-based index of the
16 item to remove.

17 Queue class (System.Collections)

18 RemoveAt

19 20 21 *Description*

22 Represents a first-in, first-out collection of objects.

23 Queues are useful for storing messages in the order they were received for
24 sequential processing. This class implements a queue as a circular array. Objects
25 stored in a **System.Collections.Queue** are inserted at one end and removed from

the other. If the number of elements added to the **System.Collections.Queue** reaches the current capacity, the capacity is automatically increased to accommodate more elements. The capacity can be decreased by calling **System.Collections.Queue.TrimToSize** .

Queue

Example Syntax:

RemoveAt

[C#] public Queue();

[C++] public: Queue();

[VB] Public Sub New()

[JScript] public function Queue(); Initializes a new instance of the

System.Collections.Queue class.

Description

Initializes a new instance of the **System.Collections.Queue** class that is empty, has the default initial capacity and uses the default growth factor.

The initial capacity is the starting capacity of the new **System.Collections.Queue** . The growth factor is the number by which the current capacity is multiplied when a greater capacity is required. The default initial capacity is 32 and the default growth factor is 2.0.

Queue

Example Syntax:

RemoveAt


```

1
2 [C#] public Queue(ICollection col);
3 [C++] public: Queue(ICollection* col);
4 [VB] Public Sub New(ByVal col As ICollection)
5 [JScript] public function Queue(col : ICollection);
6

```

7 *Description*

8 Initializes a new instance of the **System.Collections.Queue** class that
9 contains elements copied from the specified collection, has the same initial
10 capacity as the number of elements copied and uses the default growth factor.

11 The initial capacity is the starting capacity of the new
12 **System.Collections.Queue** . The growth factor is the number by which the current
13 capacity is multiplied when a greater capacity is required. The default initial
14 capacity is 32 and the default growth factor is 2.0. The
15 **System.Collections.ICollection** to copy elements from.

16 Queue

17 *Example Syntax:*

18 RemoveAt

```

19
20 [C#] public Queue(int capacity);
21 [C++] public: Queue(int capacity);
22 [VB] Public Sub New(ByVal capacity As Integer)
23 [JScript] public function Queue(capacity : int);
24

```

25 *Description*

1 Initializes a new instance of the **System.Collections.Queue** class that is
2 empty, has the specified initial capacity and uses the default growth factor.

3 The initial capacity is the starting capacity of the new
4 **System.Collections.Queue** . The growth factor is the number by which the current
5 capacity is multiplied when a greater capacity is required. The default initial
6 capacity is 32 and the default growth factor is 2.0. The initial number of elements
7 that the **System.Collections.Queue** can contain.

8 Queue

9 *Example Syntax:*

10 RemoveAt

11
12 [C#] public Queue(int capacity, float growFactor);

13 [C++] public: Queue(int capacity, float growFactor);

14 [VB] Public Sub New(ByVal capacity As Integer, ByVal growFactor As Single)

15 [JScript] public function Queue(capacity : int, growFactor : float);

16
17 *Description*

18 Initializes a new instance of the **System.Collections.Queue** class that is
19 empty, has the specified initial capacity and uses the specified growth factor.

20 The initial capacity is the starting capacity of the new
21 **System.Collections.Queue** . The growth factor is the number by which the current
22 capacity is multiplied when a greater capacity is required. The default initial
23 capacity is 32 and the default growth factor is 2.0. The initial number of elements
24 that the **System.Collections.Queue** can contain. The factor by which the capacity
25 of the **System.Collections.Queue** is expanded.

Count

RemoveAt

[C#] public virtual int Count {get;}

[C++] public: __property virtual int get_Count();

[VB] Overridable Public ReadOnly Property Count As Integer

[JScript] public function get Count() : int;

Description

Gets the number of elements contained in the **System.Collections.Queue**.

System.Collections.Queue.Count is the number of elements that are actually in the **System.Collections.Queue**. The capacity of a **System.Collections.Queue** is the number of elements that the **System.Collections.Queue** is capable of storing.

IsSynchronized

RemoveAt

[C#] public virtual bool IsSynchronized {get;}

[C++] public: __property virtual bool get_IsSynchronized();

[VB] Overridable Public ReadOnly Property IsSynchronized As Boolean

[JScript] public function get IsSynchronized() : Boolean;

Description

Gets a value indicating whether access to the **System.Collections.Queue** is synchronized (thread-safe).

To guarantee the thread safety of the **System.Collections.Queue** , all operations must be done through the wrapper returned by the **System.Collections.Queue.Synchronized(System.Collections.Queue)** method.

SyncRoot

RemoveAt

[C#] public virtual object SyncRoot {get;}

[C++] public: __property virtual Object* get_SyncRoot();

[VB] Overridable Public ReadOnly Property SyncRoot As Object

[JScript] public function get SyncRoot() : Object;

Description

Gets an object that can be used to synchronize access to the **System.Collections.Queue** .

To create a synchronized version of the **System.Collections.Queue** , use the **System.Collections.Queue.Synchronized(System.Collections.Queue)** method. However, derived classes can provide their own synchronized version of the **System.Collections.Queue** using the **System.Collections.Queue.SyncRoot** property. The synchronizing code must perform operations on the **System.Collections.Queue.SyncRoot** of the **System.Collections.Queue** , not directly on the **System.Collections.Queue** . This ensures proper operation of collections that are derived from other objects. Specifically, it maintains proper synchronization with other threads that might be simultaneously modifying the **System.Collections.Queue** object.

Clear

1
2 [C#] public virtual void Clear();
3 [C++] public: virtual void Clear();
4 [VB] Overridable Public Sub Clear()
5 [JScript] public function Clear();
6

7 *Description*

8 Removes all objects from the **System.Collections.Queue** .

9 **System.Collections.Queue.Count** is set to zero. To reset the capacity of
10 the **System.Collections.Queue** , call **System.Collections.Queue.TrimToSize** .

11 Trimming an empty **System.Collections.Queue** sets the capacity of the
12 **System.Collections.Queue** to the default capacity, not zero.

13 *Clone*

14
15 [C#] public virtual object Clone();
16 [C++] public: virtual Object* Clone();
17 [VB] Overridable Public Function Clone() As Object
18 [JScript] public function Clone() : Object;
19

20 *Description*

21 Creates a shallow copy of the **System.Collections.Queue** .

22 *Return Value:* A shallow copy of the **System.Collections.Queue** .

23 A shallow copy of a collection is a new collection containing references to
24 the same elements as the original collection. The elements themselves or anything
25 referenced by the elements are not copied. In contrast, a deep copy of a collection

copies the elements and everything directly or indirectly referenced by the elements.

Contains

[C#] public virtual bool Contains(object obj);

[C++] public: virtual bool Contains(Object* obj);

[VB] Overridable Public Function Contains(ByVal obj As Object) As Boolean

[JScript] public function Contains(obj : Object) : Boolean;

Description

Determines whether an element is in the **System.Collections.Queue** .

Return Value: **true** if *obj* is found in the **System.Collections.Queue** ; otherwise, **false** .

This method performs a linear search; therefore, the average execution time is proportional to **System.Collections.Queue.Count** . That is, this method is an $O(n)$ operation, where n is **System.Collections.Queue.Count** . The **System.Object** to locate in the **System.Collections.Queue**. The element to locate can be **null**.

CopyTo

[C#] public virtual void CopyTo(Array array, int index);

[C++] public: virtual void CopyTo(Array* array, int index);

[VB] Overridable Public Sub CopyTo(ByVal array As Array, ByVal index As Integer)

[JScript] public function CopyTo(array : Array, index : int);

Description

Copies the **System.Collections.Queue** elements to an existing one-dimensional **System.Array** , starting at the specified array index.

The elements are copied to the **System.Array** in the same order in which the enumerator iterates through the **System.Collections.Queue** . The one-dimensional **System.Array** that is the destination of the elements copied from **System.Collections.Queue**. The **System.Array** must have zero-based indexing. The zero-based index in *array* at which copying begins.

Dequeue

[C#] public virtual object Dequeue();

[C++] public: virtual Object* Dequeue();

[VB] Overridable Public Function Dequeue() As Object

[JScript] public function Dequeue() : Object;

Description

Removes and returns the object at the beginning of the **System.Collections.Queue** .

Return Value: The object that is removed from the beginning of the **System.Collections.Queue** .

This method is similar to the **System.Collections.Queue.Peek** method, but **System.Collections.Queue.Peek** does not modify the **System.Collections.Queue** .

Enqueue

```

1
2 [C#] public virtual void Enqueue(object obj);
3 [C++] public: virtual void Enqueue(Object* obj);
4 [VB] Overridable Public Sub Enqueue(ByVal obj As Object)
5 [JScript] public function Enqueue(obj : Object);
6

```

Description

Adds an object to the end of the **System.Collections.Queue** .

If **System.Collections.Queue.Count** already equals the capacity of the **System.Collections.Queue** , the capacity is increased by automatically reallocating the internal array before copying the old elements and adding the new element. The new capacity is determined by multiplying the current capacity by the growth factor, which is determined when the **System.Collections.Queue** is constructed. The object to add to the **System.Collections.Queue**.

GetEnumerator

```

17 [C#] public virtual IEnumerator GetEnumerator();
18 [C++] public: virtual IEnumerator* GetEnumerator();
19 [VB] Overridable Public Function GetEnumerator() As IEnumerator
20 [JScript] public function GetEnumerator() : IEnumerator;
21

```

Description

Returns an enumerator that can iterate through the **System.Collections.Queue** .

1 *Return Value:* An **System.Collections.IEnumerator** for the
2 **System.Collections.Queue** .

3 Enumerators are intended to be used only to read data in the collection.

4 Enumerators cannot be used to modify the underlying collection.

5 Peek

6
7 [C#] public virtual object Peek();

8 [C++] public: virtual Object* Peek();

9 [VB] Overridable Public Function Peek() As Object

10 [JScript] public function Peek() : Object;

11
12 *Description*

13 Returns the object at the beginning of the **System.Collections.Queue**
14 without removing it.

15 *Return Value:* The object at the beginning of the **System.Collections.Queue** .

16 This method is similar to the **System.Collections.Queue.Dequeue** method,
17 but **System.Collections.Queue.Peek** does not modify the
18 **System.Collections.Queue** .

19 Synchronized

20
21 [C#] public static Queue Synchronized(Queue queue);

22 [C++] public: static Queue* Synchronized(Queue* queue);

23 [VB] Public Shared Function Synchronized(ByVal queue As Queue) As Queue

24 [JScript] public static function Synchronized(queue : Queue) : Queue;

1
2 *Description*

3 Returns a **System.Collections.Queue** wrapper that is synchronized (thread-
4 safe).

5 *Return Value:* A **System.Collections.Queue** wrapper that is synchronized (thread-
6 safe).

7 To guarantee the thread safety of the **System.Collections.Queue** , all
8 operations must be done through this wrapper only. The
9 **System.Collections.Queue** to synchronize.

10 **ToArray**

11
12 [C#] public virtual object[] ToArray();

13 [C++] public: virtual Object* ToArray() __gc[];

14 [VB] Overridable Public Function ToArray() As Object()

15 [JScript] public function ToArray() : Object[];

16
17 *Description*

18 Copies the **System.Collections.Queue** elements to a new array.

19 *Return Value:* A new array containing elements copied from the
20 **System.Collections.Queue** .

21 The **System.Collections.Queue** is not modified. The order of the elements
22 in the new array is the same as the order of the elements from the beginning of the
23 **System.Collections.Queue** to its end.

24 **TrimToSize**

1
2 [C#] public virtual void TrimToSize();

3 [C++] public: virtual void TrimToSize();

4 [VB] Overridable Public Sub TrimToSize()

5 [JScript] public function TrimToSize();

6
7 *Description*

8 Sets the capacity to the actual number of elements in the

9 **System.Collections.Queue** .

10 This method can be used to minimize a list's memory overhead if no new
11 elements will be added to the list.

12 ReadOnlyCollectionBase class (System.Collections)

13 TrimToSize

14
15
16 *Description*

17 Provides the **abstract** base class for a strongly typed read-only collection.

18 A **System.Collections.ReadOnlyCollectionBase** instance is always read-
19 only. See **System.Collections.CollectionBase** for a modifiable version of this
20 class.

21 ReadOnlyCollectionBase

22 *Example Syntax:*

23 TrimToSize

24
25 [C#] protected ReadOnlyCollectionBase();

```

1  [C++] protected: ReadOnlyCollectionBase();
2  [VB] Protected Sub New()
3  [JScript] protected function ReadOnlyCollectionBase();
4      Count
5      TrimToSize
6
7  [C#] public int Count {get;}
8  [C++] public: __property int get_Count();
9  [VB] Public ReadOnly Property Count As Integer
10 [JScript] public function get Count() : int;
11
12 Description
13     Gets the number of elements contained in the
14     System.Collections.ReadOnlyCollectionBase instance.
15     InnerList
16     TrimToSize
17
18 [C#] protected ArrayList InnerList {get;}
19 [C++] protected: __property ArrayList* get_InnerList();
20 [VB] Protected ReadOnly Property InnerList As ArrayList
21 [JScript] protected function get InnerList() : ArrayList;
22
23 Description
24     Gets the list of elements contained in the
25     System.Collections.ReadOnlyCollectionBase instance.

```

GetEnumerator

[C#] public IEnumerator GetEnumerator();

[C++] public: __sealed IEnumerator* GetEnumerator();

[VB] NotOverridable Public Function GetEnumerator() As IEnumerator

[JScript] public function GetEnumerator() : IEnumerator;

Description

Returns an enumerator that can iterate through the **System.Collections.ReadOnlyCollectionBase** instance.

Return Value: An **System.Collections.IEnumerator** for the **System.Collections.ReadOnlyCollectionBase** instance.

Enumerators are intended to be used only to read data in the collection.

Enumerators cannot be used to modify the underlying collection.

ICollection.CopyTo

[C#] void ICollection.CopyTo(Array array, int index);

[C++] void ICollection::CopyTo(Array* array, int index);

[VB] Sub CopyTo(ByVal array As Array, ByVal index As Integer) Implements
ICollection.CopyTo

[JScript] function ICollection.CopyTo(array : Array, index : int);

SortedList class (System.Collections)

ToString

Description

Represents a collection of key-and-value pairs that are sorted by the keys and are accessible by key and by index.

A **System.Collections.SortedList** is a hybrid between a **System.Collections.Hashtable** and an **System.Array** . When an element is accessed by its key using the **System.Collections.SortedList.Item(System.Object)** indexer property, it behaves like a **System.Collections.Hashtable** . When an element is accessed by its index using **System.Collections.SortedList.GetByIndex(System.Int32)** or **System.Collections.SortedList.SetByIndex(System.Int32,System.Object)** , it behaves like an **System.Array** .

SortedList

Example Syntax:

ToString

[C#] public SortedList();

[C++] public: SortedList();

[VB] Public Sub New()

[JScript] public function SortedList(); Initializes a new instance of the

System.Collections.SortedList class.

Description

1 Initializes a new instance of the **System.Collections.SortedList** class that
2 is empty, has the default initial capacity and is sorted according to the
3 **System.IComparable** interface implemented by each key added to the
4 **System.Collections.SortedList** .

5 The initial capacity is the starting capacity of the new
6 **System.Collections.SortedList** . The default initial capacity for a
7 **System.Collections.SortedList** is 16.

8 SortedList

9 *Example Syntax:*

10 ToString

11
12 [C#] public SortedList(IComparer comparer);

13 [C++] public: SortedList(IComparer* comparer);

14 [VB] Public Sub New(ByVal comparer As IComparer)

15 [JScript] public function SortedList(comparer : IComparer);

16
17 *Description*

18 Initializes a new instance of the **System.Collections.SortedList** class that
19 is empty, has the default initial capacity and is sorted according to the specified
20 **System.Collections.IComparer** interface.

21 The initial capacity is the starting capacity of the new
22 **System.Collections.SortedList** . The default initial capacity for a
23 **System.Collections.SortedList** is 16. The **System.Collections.IComparer**
24 implementation to use when comparing keys.

25 SortedList

Example Syntax:

ToString

[C#] public SortedList(IDictionary d);

[C++] public: SortedList(IDictionary* d);

[VB] Public Sub New(ByVal d As IDictionary)

[JScript] public function SortedList(d : IDictionary);

Description

Initializes a new instance of the **System.Collections.SortedList** class that contains elements copied from the specified dictionary, has the same initial capacity as the number of elements copied and is sorted according to the **System.IComparable** interface implemented by each key.

The initial capacity is the starting capacity of the new **System.Collections.SortedList**. When adding elements to the list, if the number of elements exceeds the current capacity, the capacity is automatically doubled. The **System.Collections.IDictionary** to copy to a new **System.Collections.SortedList**.

SortedList

Example Syntax:

ToString

[C#] public SortedList(int initialCapacity);

[C++] public: SortedList(int initialCapacity);

[VB] Public Sub New(ByVal initialCapacity As Integer)

1 [JScript] public function SortedList(initialCapacity : int);

3 *Description*

4 Initializes a new instance of the **System.Collections.SortedList** class that
5 is empty, has the specified initial capacity and is sorted according to the
6 **System.IComparable** interface implemented by each key added to the
7 **System.Collections.SortedList** .

8 The initial capacity is the starting capacity of the new
9 **System.Collections.SortedList** . The default initial capacity for a
10 **System.Collections.SortedList** is 16. The initial number of elements that the
11 **System.Collections.SortedList** can contain.

12 SortedList

13 *Example Syntax:*

14 ToString

16 [C#] public SortedList(IComparer comparer, int capacity);

17 [C++] public: SortedList(IComparer* comparer, int capacity);

18 [VB] Public Sub New(ByVal comparer As IComparer, ByVal capacity As Integer)

19 [JScript] public function SortedList(comparer : IComparer, capacity : int);

21 *Description*

22 Initializes a new instance of the **System.Collections.SortedList** class that
23 is empty, has the specified initial capacity and is sorted according to the specified
24 **System.Collections.IComparer** interface.

The initial capacity is the starting capacity of the new **System.Collections.SortedList**. The default initial capacity for a **System.Collections.SortedList** is 16. The **System.Collections.IComparer** implementation to use when comparing keys. The initial number of elements that the **System.Collections.SortedList** can contain.

SortedList

Example Syntax:

ToString

[C#] public SortedList(IDictionary d, IComparer comparer);

[C++] public: SortedList(IDictionary* d, IComparer* comparer);

[VB] Public Sub New(ByVal d As IDictionary, ByVal comparer As IComparer)

[JScript] public function SortedList(d : IDictionary, comparer : IComparer);

Description

Initializes a new instance of the **System.Collections.SortedList** class that contains elements copied from the specified dictionary, has the same initial capacity as the number of elements copied and is sorted according to the specified **System.Collections.IComparer** interface.

The initial capacity is the starting capacity of the new **System.Collections.SortedList**. When adding elements to the list, if the number of elements exceeds the current capacity, the capacity is automatically doubled. The **System.Collections.IDictionary** to copy to a new **System.Collections.SortedList**. The **System.Collections.IComparer** implementation to use when comparing keys.

Capacity

ToString

[C#] public virtual int Capacity {get; set;}

[C++] public: __property virtual int get_Capacity();public: __property virtual void
set_Capacity(int);

[VB] Overridable Public Property Capacity As Integer

[JScript] public function get Capacity() : int;public function set Capacity(int);

Description

Gets or sets the capacity of the **System.Collections.SortedList** .

If the number of elements added to the list reaches the current capacity, the
capacity is automatically doubled.

Count

ToString

[C#] public virtual int Count {get;}

[C++] public: __property virtual int get_Count();

[VB] Overridable Public ReadOnly Property Count As Integer

[JScript] public function get Count() : int;

Description

Gets the number of elements contained in the
System.Collections.SortedList .

Each element is a key-and-value pair that can be accessed as a **System.Collections.DictionaryEntry** object.

IsFixedSize

ToString

[C#] public virtual bool IsFixedSize {get;}

[C++] public: __property virtual bool get_IsFixedSize();

[VB] Overridable Public ReadOnly Property IsFixedSize As Boolean

[JScript] public function get IsFixedSize() : Boolean;

Description

Gets a value indicating whether the **System.Collections.SortedList** has a fixed size.

A collection with a fixed size does not allow the addition or removal of elements, but it allows the modification of existing elements.

IsReadOnly

ToString

[C#] public virtual bool IsReadOnly {get;}

[C++] public: __property virtual bool get_IsReadOnly();

[VB] Overridable Public ReadOnly Property IsReadOnly As Boolean

[JScript] public function get IsReadOnly() : Boolean;

Description

Gets a value indicating whether the **System.Collections.SortedList** is read-only.

IsSynchronized

ToString

[C#] public virtual bool IsSynchronized {get;}

[C++] public: __property virtual bool get_IsSynchronized();

[VB] Overridable Public ReadOnly Property IsSynchronized As Boolean

[JScript] public function get IsSynchronized() : Boolean;

Description

Gets a value indicating whether access to the **System.Collections.SortedList** is synchronized (thread-safe).

To guarantee the thread safety of the **System.Collections.SortedList**, all operations must be done through the wrapper returned by the **System.Collections.SortedList.Synchronized(System.Collections.SortedList)** method.

Item

ToString

[C#] public virtual object this[object key] {get; set;}

[C++] public: __property virtual Object* get_Item(Object* key);public:

__property virtual void set_Item(Object* key, Object*);

[VB] Overridable Public Default Property Item(ByVal key As Object) As Object

[JScript] returnValue = SortedListObject.Item(key);SortedListObject.Item(key) =

returnValue;

Description

Gets and sets the value associated with a specific key in the

System.Collections.SortedList .

If setting the value of *key* and *key* does not exist in the

System.Collections.SortedList , a new element is created with the specified key and the specified value. The key associated with the value to get or set.

Keys

ToString

[C#] public virtual ICollection Keys {get;}

[C++] public: __property virtual ICollection* get_Keys();

[VB] Overridable Public ReadOnly Property Keys As ICollection

[JScript] public function get Keys() : ICollection;

Description

Gets the keys in the **System.Collections.SortedList** .

The **System.Collections.ICollection** is a read-only view of the keys of the **System.Collections.SortedList** . Modifications made to the underlying **System.Collections.SortedList** are immediately reflected in the **System.Collections.ICollection** .

SyncRoot

ToString

```

1 [C#] public virtual object SyncRoot {get;}
2
3 [C++] public: __property virtual Object* get_SyncRoot();
4
5 [VB] Overridable Public ReadOnly Property SyncRoot As Object
6
7 [JScript] public function get SyncRoot() : Object;
8

```

Description

Gets an object that can be used to synchronize access to the

System.Collections.SortedList .

To create a synchronized version of the **System.Collections.SortedList** , use the

System.Collections.SortedList.Synchronized(System.Collections.SortedList)

method. However, derived classes can provide their own synchronized version of the **System.Collections.SortedList** using the

System.Collections.SortedList.SyncRoot property. The synchronizing code must perform operations on the **System.Collections.SortedList.SyncRoot** of the

System.Collections.SortedList , not directly on the

System.Collections.SortedList . This ensures proper operation of collections that

are derived from other objects. Specifically, it maintains proper synchronization with other threads that might be simultaneously modifying the

System.Collections.SortedList object.

Values

ToString

```

22
23
24
25 [C#] public virtual ICollection Values {get;}

```

1 [C++] public: __property virtual ICollection* get_Values();

2 [VB] Overridable Public ReadOnly Property Values As ICollection

3 [JScript] public function get Values() : ICollection;

4
5 *Description*

6 Gets the values in the **System.Collections.SortedList** .

7 The **System.Collections.ICollection** is a read-only view of the values of
8 the **System.Collections.SortedList** . Modifications made to the underlying
9 **System.Collections.SortedList** are immediately reflected in the
10 **System.Collections.ICollection** .

11 **Add**

12
13 [C#] public virtual void Add(object key, object value);

14 [C++] public: virtual void Add(Object* key, Object* value);

15 [VB] Overridable Public Sub Add(ByVal key As Object, ByVal value As Object)

16 [JScript] public function Add(key : Object, value : Object);

17
18 *Description*

19 Adds an element with the specified key and value to the
20 **System.Collections.SortedList** .

21 If the number of elements added to the list reaches the current capacity, the
22 capacity is automatically doubled. The insertion point is determined based on the
23 comparer selected, either explicitly or by default when the
24 **System.Collections.SortedList** was created. The key of the element to add. The
25 value of the element to add.

Clear

[C#] public virtual void Clear();

[C++] public: virtual void Clear();

[VB] Overridable Public Sub Clear()

[JScript] public function Clear();

Description

Removes all elements from the **System.Collections.SortedList** .

System.Collections.SortedList.Count is set to zero.

Clone

[C#] public virtual object Clone();

[C++] public: virtual Object* Clone();

[VB] Overridable Public Function Clone() As Object

[JScript] public function Clone() : Object;

Description

Creates a shallow copy of the **System.Collections.SortedList** .

Return Value: A shallow copy of the **System.Collections.SortedList** .

A shallow copy of a collection is a new collection containing references to the same elements as the original collection. The elements themselves or anything referenced by the elements are not copied. In contrast, a deep copy of a collection copies the elements and everything directly or indirectly referenced by the elements.

Contains

[C#] public virtual bool Contains(object key);
[C++] public: virtual bool Contains(Object* key);
[VB] Overridable Public Function Contains(ByVal key As Object) As Boolean
[JScript] public function Contains(key : Object) : Boolean;

Description

Determines whether the **System.Collections.SortedList** contains a specific key.

Return Value: **true** if the **System.Collections.SortedList** contains an element with the specified *key* ; otherwise, **false** .

The elements of a **System.Collections.SortedList** are sorted by the keys either according to a specific **System.Collections.IComparer** implementation specified when the **System.Collections.SortedList** is created or according to the **System.IComparable** implementation provided by the keys themselves. The key to locate in the **System.Collections.SortedList**.

ContainsKey

[C#] public virtual bool ContainsKey(object key);
[C++] public: virtual bool ContainsKey(Object* key);
[VB] Overridable Public Function ContainsKey(ByVal key As Object) As Boolean
[JScript] public function ContainsKey(key : Object) : Boolean;

1
2 *Description*

3 Determines whether the **System.Collections.SortedList** contains a specific
4 key.

5 *Return Value:* **true** if the **System.Collections.SortedList** contains an element
6 with the specified *key* ; otherwise, **false** .

7 The elements of a **System.Collections.SortedList** are sorted by the keys
8 either according to a specific **System.Collections.IComparer** implementation
9 specified when the **System.Collections.SortedList** is created or according to the
10 **System.IComparable** implementation provided by the keys themselves. The key
11 to locate in the **System.Collections.SortedList**.

12 ContainsValue

13
14 [C#] public virtual bool ContainsValue(object value);

15 [C++] public: virtual bool ContainsValue(Object* value);

16 [VB] Overridable Public Function ContainsValue(ByVal value As Object) As

17 Boolean

18 [JScript] public function ContainsValue(value : Object) : Boolean;

19
20 *Description*

21 Determines whether the **System.Collections.SortedList** contains a specific
22 value.

23 *Return Value:* **true** if the **System.Collections.SortedList** contains an element
24 with the specified *value* ; otherwise, **false** .
25

This method performs a linear search; therefore, the average execution time is proportional to **System.Collections.SortedList.Count** . That is, this method is an $O(n)$ operation, where n is **System.Collections.SortedList.Count** . The value to locate in the **System.Collections.SortedList**.

CopyTo

[C#] public virtual void CopyTo(Array array, int arrayIndex);

[C++] public: virtual void CopyTo(Array* array, int arrayIndex);

[VB] Overridable Public Sub CopyTo(ByVal array As Array, ByVal arrayIndex As Integer)

[JScript] public function CopyTo(array : Array, arrayIndex : int);

Description

Copies the **System.Collections.SortedList** elements to a one-dimensional **System.Array** instance at the specified index.

The key-and-value pairs are copied to the **System.Array** in the same order in which the enumerator iterates through the **System.Collections.SortedList** . The one-dimensional **System.Array** that is the destination of the **System.Collections.DictionaryEntry** objects copied from **System.Collections.SortedList**. The **System.Array** must have zero-based indexing. The zero-based index in *array* at which copying begins.

GetByIndex

[C#] public virtual object GetByIndex(int index);

[C++] public: virtual Object* GetByIndex(int index);

1 [VB] Overridable Public Function GetByIndex(ByVal index As Integer) As

2 Object

3 [JScript] public function GetByIndex(index : int) : Object;

4
5 *Description*

6 Gets the value at the specified index of the **System.Collections.SortedList**

7 .
8 *Return Value:* The value at the specified index of the

9 **System.Collections.SortedList** .

10 The index sequence is based on the sort sequence. When an element is
11 added, it is inserted into **System.Collections.SortedList** in the correct sort order,
12 and the indexing adjusts accordingly. When an element removed, the indexing
13 also adjusts accordingly. Therefore, the index of a specific key-and-value pair
14 might change as elements are added or removed from the
15 **System.Collections.SortedList** . The zero-based index of the value to get.

16 GetEnumerator

17
18 [C#] public virtual IDictionaryEnumerator GetEnumerator();

19 [C++] public: virtual IDictionaryEnumerator* GetEnumerator();

20 [VB] Overridable Public Function GetEnumerator() As IDictionaryEnumerator

21 [JScript] public function GetEnumerator() : IDictionaryEnumerator;

22
23 *Description*

24 Returns an enumerator that can iterate through the

25 **System.Collections.SortedList** .

1 *Return Value:* An **System.Collections.IDictionaryEnumerator** for the
2 **System.Collections.SortedList** .

3 Enumerators are intended to be used only to read data in the collection.

4 Enumerators cannot be used to modify the underlying collection.

5 **GetKey**

6
7 [C#] public virtual object GetKey(int index);

8 [C++] public: virtual Object* GetKey(int index);

9 [VB] Overridable Public Function GetKey(ByVal index As Integer) As Object

10 [JScript] public function GetKey(index : int) : Object;

11
12 *Description*

13 Gets the key at the specified index of the **System.Collections.SortedList** .

14 *Return Value:* The key at the specified index of the

15 **System.Collections.SortedList** .

16 The index sequence is based on the sort sequence. When an element is
17 added, it is inserted into **System.Collections.SortedList** in the correct sort order,
18 and the indexing adjusts accordingly. When an element removed, the indexing
19 also adjusts accordingly. Therefore, the index of a specific key-and-value pair
20 might change as elements are added or removed from the
21 **System.Collections.SortedList** . The zero-based index of the key to get.

22 **GetKeyList**

23
24 [C#] public virtual IList GetKeyList();

25 [C++] public: virtual IList* GetKeyList();

1 [VB] Overridable Public Function GetKeyList() As IList

2 [JScript] public function GetKeyList() : IList;

3
4 *Description*

5 Gets the keys in the **System.Collections.SortedList** .

6 *Return Value:* An **System.Collections.IList** containing the keys in the

7 **System.Collections.SortedList** .

8 The returned **System.Collections.IList** is a read-only view of the keys of
9 the **System.Collections.SortedList** . Modifications made to the underlying
10 **System.Collections.SortedList** are immediately reflected in the
11 **System.Collections.IList** .

12 GetValueList

13
14 [C#] public virtual IList GetValueList();

15 [C++] public: virtual IList* GetValueList();

16 [VB] Overridable Public Function GetValueList() As IList

17 [JScript] public function GetValueList() : IList;

18
19 *Description*

20 Gets the values in the **System.Collections.SortedList** .

21 *Return Value:* An **System.Collections.IList** containing the values in the

22 **System.Collections.SortedList** .

23 The returned **System.Collections.IList** is a read-only view of the values of
24 the **System.Collections.SortedList** . Modifications made to the underlying
25

1 **System.Collections.SortedList** are immediately reflected in the

2 **System.Collections.IList** .

3 **IndexOfKey**

4
5 [C#] public virtual int IndexOfKey(object key);

6 [C++] public: virtual int IndexOfKey(Object* key);

7 [VB] Overridable Public Function IndexOfKey(ByVal key As Object) As Integer

8 [JScript] public function IndexOfKey(key : Object) : int;

9
10 *Description*

11 Returns the zero-based index of the specified key in the

12 **System.Collections.SortedList** .

13 *Return Value:* The zero-based index of *key* , if *key* is found in the

14 **System.Collections.SortedList** ; otherwise, -1.

15 The elements of a **System.Collections.SortedList** are sorted by the keys
16 either according to a specific **System.Collections.IComparer** implementation
17 specified when the **System.Collections.SortedList** is created or according to the
18 **System.IComparable** implementation provided by the keys themselves. The key
19 to locate in the **System.Collections.SortedList**.

20 **IndexOfValue**

21
22 [C#] public virtual int IndexOfValue(object value);

23 [C++] public: virtual int IndexOfValue(Object* value);

24 [VB] Overridable Public Function IndexOfValue(ByVal value As Object) As

25 Integer

1 [JScript] public function IndexOfValue(value : Object) : int;

3 *Description*

4 Returns the zero-based index of the first occurrence of the specified value
5 in the **System.Collections.SortedList** .

6 *Return Value:* The zero-based index of the first occurrence of *value* , if *value* is
7 found in the **System.Collections.SortedList** ; otherwise, -1.

8 The index sequence is based on the sort sequence. When an element is
9 added, it is inserted into **System.Collections.SortedList** in the correct sort order,
10 and the indexing adjusts accordingly. When an element removed, the indexing
11 also adjusts accordingly. Therefore, the index of a specific key-and-value pair
12 might change as elements are added or removed from the
13 **System.Collections.SortedList** . The value to locate in the
14 **System.Collections.SortedList**.

15 Remove

17 [C#] public virtual void Remove(object key);

18 [C++] public: virtual void Remove(Object* key);

19 [VB] Overridable Public Sub Remove(ByVal key As Object)

20 [JScript] public function Remove(key : Object);

22 *Description*

23 Removes the element with the specified key from
24 **System.Collections.SortedList** .

1 If the **System.Collections.SortedList** does not contain an element with the
2 specified key, the **System.Collections.SortedList** remains unchanged. No
3 exception is thrown. The key of the element to remove.

4 RemoveAt

5
6 [C#] public virtual void RemoveAt(int index);

7 [C++] public: virtual void RemoveAt(int index);

8 [VB] Overridable Public Sub RemoveAt(ByVal index As Integer)

9 [JScript] public function RemoveAt(index : int);

10 11 *Description*

12 Removes the element at the specified index of
13 **System.Collections.SortedList** .

14 The index sequence is based on the sort sequence. When an element is
15 added, it is inserted into **System.Collections.SortedList** in the correct sort order,
16 and the indexing adjusts accordingly. When an element removed, the indexing
17 also adjusts accordingly. Therefore, the index of a specific key-and-value pair
18 might change as elements are added or removed from the
19 **System.Collections.SortedList** . The zero-based index of the element to remove.

20 SetByIndex

21
22 [C#] public virtual void SetByIndex(int index, object value);

23 [C++] public: virtual void SetByIndex(int index, Object* value);

24 [VB] Overridable Public Sub SetByIndex(ByVal index As Integer, ByVal value
25 As Object)

1 [JScript] public function SetByIndex(index : int, value : Object);

3 *Description*

4 Replaces the value at a specific index in the

5 **System.Collections.SortedList** .

6 The index sequence is based on the sort sequence. When an element is
7 added, it is inserted into **System.Collections.SortedList** in the correct sort order,
8 and the indexing adjusts accordingly. When an element removed, the indexing
9 also adjusts accordingly. Therefore, the index of a specific key-and-value pair
10 might change as elements are added or removed from the

11 **System.Collections.SortedList** . The zero-based index at which to save *value*.

12 The **System.Object** to save into the **System.Collections.SortedList**.

13 Synchronized

15 [C#] public static SortedList Synchronized(SortedList list);

16 [C++] public: static SortedList* Synchronized(SortedList* list);

17 [VB] Public Shared Function Synchronized(ByVal list As SortedList) As
18 SortedList

19 [JScript] public static function Synchronized(list : SortedList) : SortedList;

21 *Description*

22 Returns a synchronized (thread-safe) wrapper for the

23 **System.Collections.SortedList** .

24 *Return Value:* A synchronized (thread-safe) wrapper for the

25 **System.Collections.SortedList** .

To guarantee the thread safety of the **System.Collections.SortedList** , all operations must be done through this wrapper only. The **System.Collections.SortedList** to synchronize.

IEnumerable.GetEnumerator

[C#] **IEnumerator IEnumerable.GetEnumerator();**

[C++] **IEnumerator* IEnumerable::GetEnumerator();**

[VB] **Function GetEnumerator() As IEnumerator Implements**

IEnumerable.GetEnumerator

[JScript] **function IEnumerable.GetEnumerator() : IEnumerator;**

TrimToSize

[C#] **public virtual void TrimToSize();**

[C++] **public: virtual void TrimToSize();**

[VB] **Overridable Public Sub TrimToSize()**

[JScript] **public function TrimToSize();**

Description

Sets the capacity to the actual number of elements in the **System.Collections.SortedList** .

This method can be used to minimize a list's memory overhead if no new elements will be added to the list.

Stack class (System.Collections)

TrimToSize

1
2
3 *Description*

4 Represents a simple last-in-first-out collection of objects.

5 **System.Collections.Stack** is implemented as a circular buffer.

6 **Stack**

7 *Example Syntax:*

8 TrimToSize

9
10 [C#] public Stack();

11 [C++] public: Stack();

12 [VB] Public Sub New()

13 [JScript] public function Stack(); Initializes a new instance of the

14 **System.Collections.Stack** class.

15
16 *Description*

17 Initializes a new instance of the **System.Collections.Stack** class that is
18 empty and has the default initial capacity.

19 The initial capacity is the starting capacity of the new
20 **System.Collections.Stack** . The default initial capacity for a
21 **System.Collections.Stack** is 10.

22 **Stack**

23 *Example Syntax:*

24 TrimToSize
25

```

1
2 [C#] public Stack(ICollection col);
3 [C++] public: Stack(ICollection* col);
4 [VB] Public Sub New(ByVal col As ICollection)
5 [JScript] public function Stack(col : ICollection);
6

```

7 *Description*

8 Initializes a new instance of the **System.Collections.Stack** class that
9 contains elements copied from the specified collection and has the same initial
10 capacity as the number of elements copied.

11 The initial capacity is the starting capacity of the new
12 **System.Collections.Stack** . If the number of elements added to the stack reaches
13 the current capacity, the capacity is automatically doubled. The
14 **System.Collections.ICollection** to copy elements from.

15 Stack

16 *Example Syntax:*

17 TrimToSize

```

18
19 [C#] public Stack(int initialCapacity);
20 [C++] public: Stack(int initialCapacity);
21 [VB] Public Sub New(ByVal initialCapacity As Integer)
22 [JScript] public function Stack(initialCapacity : int);
23

```

24 *Description*

1 Initializes a new instance of the **System.Collections.Stack** class that is
2 empty and has the specified initial capacity or the default initial capacity,
3 whichever is greater.

4 The initial capacity is the starting capacity of the new
5 **System.Collections.Stack** . The default initial capacity for a
6 **System.Collections.Stack** is 10. The initial number of elements that the
7 **System.Collections.Stack** can contain.

8 Count

9 TrimToSize

10
11 [C#] public virtual int Count {get;}

12 [C++] public: __property virtual int get_Count();

13 [VB] Overridable Public ReadOnly Property Count As Integer

14 [JScript] public function get Count() : int;

15
16 *Description*

17 Gets the number of elements contained in the **System.Collections.Stack** .

18 IsSynchronized

19 TrimToSize

20
21 [C#] public virtual bool IsSynchronized {get;}

22 [C++] public: __property virtual bool get_IsSynchronized();

23 [VB] Overridable Public ReadOnly Property IsSynchronized As Boolean

24 [JScript] public function get IsSynchronized() : Boolean;

1
2 *Description*

3 Gets a value indicating whether access to the **System.Collections.Stack** is
4 synchronized (thread-safe).

5 To guarantee the thread safety of the **System.Collections.Stack** , all
6 operations must be done through the wrapper returned by the
7 **System.Collections.Stack.Synchronized(System.Collections.Stack)** method.

8 SyncRoot

9 TrimToSize

10
11 [C#] public virtual object SyncRoot {get;}

12 [C++] public: __property virtual Object* get_SyncRoot();

13 [VB] Overridable Public ReadOnly Property SyncRoot As Object

14 [JScript] public function get SyncRoot() : Object;

15
16 *Description*

17 Gets an object that can be used to synchronize access to the
18 **System.Collections.Stack** .

19 To create a synchronized version of the **System.Collections.Stack** , use the
20 **System.Collections.Stack.Synchronized(System.Collections.Stack)** method.

21 However, derived classes can provide their own synchronized version of the
22 **System.Collections.Stack** using the **System.Collections.Stack.SyncRoot**
23 property. The synchronizing code must perform operations on the
24 **System.Collections.Stack.SyncRoot** of the **System.Collections.Stack** , not
25 directly on the **System.Collections.Stack** . This ensures proper operation of

collections that are derived from other objects. Specifically, it maintains proper synchronization with other threads that might be simultaneously modifying the **System.Collections.Stack** object.

Clear

[C#] public virtual void Clear();

[C++] public: virtual void Clear();

[VB] Overridable Public Sub Clear()

[JScript] public function Clear();

Description

Removes all objects from the **System.Collections.Stack**.

System.Collections.Stack.Count is set to zero.

Clone

[C#] public virtual object Clone();

[C++] public: virtual Object* Clone();

[VB] Overridable Public Function Clone() As Object

[JScript] public function Clone() : Object;

Description

Creates a shallow copy of the **System.Collections.Stack**.

Return Value: A shallow copy of the **System.Collections.Stack**.

A shallow copy of a collection is a new collection containing references to the same elements as the original collection. The elements themselves or anything

referenced by the elements are not copied. In contrast, a deep copy of a collection copies the elements and everything directly or indirectly referenced by the elements.

Contains

[C#] public virtual bool Contains(object obj);

[C++] public: virtual bool Contains(Object* obj);

[VB] Overridable Public Function Contains(ByVal obj As Object) As Boolean

[JScript] public function Contains(obj : Object) : Boolean;

Description

Determines whether an element is in the **System.Collections.Stack** .

Return Value: **true** if *obj* is found in the **System.Collections.Stack** ; otherwise, **false** .

This method performs a linear search; therefore, the average execution time is proportional to **System.Collections.Stack.Count** . That is, this method is an $O(n)$ operation, where n is **System.Collections.Stack.Count** . The **System.Object** to locate in the **System.Collections.Stack**. The element to locate can be **null**.

CopyTo

[C#] public virtual void CopyTo(Array array, int index);

[C++] public: virtual void CopyTo(Array* array, int index);

[VB] Overridable Public Sub CopyTo(ByVal array As Array, ByVal index As Integer)

[JScript] public function CopyTo(array : Array, index : int);

Description

Copies the **System.Collections.Stack** to an existing one-dimensional **System.Array** , starting at the specified array index.

The elements are copied onto the array in a last-in-first-out order, similar to the order of the elements returned by a succession of calls to **System.Collections.Stack.Pop** . The one-dimensional **System.Array** that is the destination of the elements copied from **System.Collections.Stack**. The **System.Array** must have zero-based indexing. The zero-based index in *array* at which copying begins.

GetEnumerator

[C#] public virtual IEnumerator GetEnumerator();

[C++] public: virtual IEnumerator* GetEnumerator();

[VB] Overridable Public Function GetEnumerator() As IEnumerator

[JScript] public function GetEnumerator() : IEnumerator;

Description

Returns an **System.Collections.IEnumerator** for the **System.Collections.Stack** .

Return Value: An **System.Collections.IEnumerator** for the **System.Collections.Stack** .

Enumerators are intended to be used only to read data in the collection. Enumerators cannot be used to modify the underlying collection.

Peek

1
2 [C#] public virtual object Peek();

3 [C++] public: virtual Object* Peek();

4 [VB] Overridable Public Function Peek() As Object

5 [JScript] public function Peek() : Object;

6
7 *Description*

8 Returns the object at the top of the **System.Collections.Stack** without
9 removing it.

10 *Return Value:* The **System.Object** at the top of the **System.Collections.Stack** .

11 **null** can be pushed onto the **System.Collections.Stack** as a placeholder, if
12 needed. To distinguish between a null value and the end of the stack, check the
13 **System.Collections.Stack.Count** property or catch the
14 **System.InvalidOperationException** , which is thrown when the
15 **System.Collections.Stack** is empty.

16 **Pop**

17
18 [C#] public virtual object Pop();

19 [C++] public: virtual Object* Pop();

20 [VB] Overridable Public Function Pop() As Object

21 [JScript] public function Pop() : Object;

22
23 *Description*

24 Removes and returns the object at the top of the **System.Collections.Stack**

1 *Return Value:* The **System.Object** removed from the top of the
2 **System.Collections.Stack** .

3 **System.Collections.Stack** is implemented as a circular buffer.

4 Push

5
6 [C#] public virtual void Push(object obj);

7 [C++] public: virtual void Push(Object* obj);

8 [VB] Overridable Public Sub Push(ByVal obj As Object)

9 [JScript] public function Push(obj : Object);

10
11 *Description*

12 Inserts an object at the top of the **System.Collections.Stack** .

13 **System.Collections.Stack** is implemented as a circular buffer. The
14 **System.Object** to push onto the **System.Collections.Stack**.

15 Synchronized

16
17 [C#] public static Stack Synchronized(Stack stack);

18 [C++] public: static Stack* Synchronized(Stack* stack);

19 [VB] Public Shared Function Synchronized(ByVal stack As Stack) As Stack

20 [JScript] public static function Synchronized(stack : Stack) : Stack;

21
22 *Description*

23 Returns a synchronized (thread-safe) wrapper for the

24 **System.Collections.Stack** .

25 *Return Value:* A synchronized wrapper around the **System.Collections.Stack** .

To guarantee the thread safety of the **System.Collections.Stack** , all operations must be done through this wrapper. The **System.Collections.Stack** to synchronize.

ToArray

[C#] public virtual object[] ToArray();

[C++] public: virtual Object* ToArray() __gc[];

[VB] Overridable Public Function ToArray() As Object()

[JScript] public function ToArray() : Object[];

Description

Copies the **System.Collections.Stack** to a new array.

Return Value: A new array containing copies of the elements of the **System.Collections.Stack** .

The elements are copied onto the array in a last-in-first-out order, similar to the order

System.Collections.Specialized

The namespace contains specialized and strongly-typed collections; for example, a linked list dictionary, a bit vector and collections that contain only strings.

Description

The **System.Collections.Specialized** namespace contains specialized and strongly-typed collections; for example, a linked list dictionary, a bit vector and collections that contain only strings.

BitVector32 structure (System.Collections.Specialized)

Description

Provides a simple structure that stores Boolean values and small integers in 32 bits of memory.

System.Collections.Specialized.BitVector32 is more efficient than **System.Collections.BitArray** for Boolean values and small integers that are used internally.

Constructors:

BitVector32

Example Syntax:

[C#] public BitVector32(BitVector32 value);

[C++] public: BitVector32(BitVector32 value);

[VB] Public Sub New(ByVal value As BitVector32)

[JScript] public function BitVector32(value : BitVector32);

Description

Initializes a new instance of the **System.Collections.Specialized.BitVector32** structure containing the data represented in an existing **System.Collections.Specialized.BitVector32** structure. A **System.Collections.Specialized.BitVector32** structure that contains the data to copy.

BitVector32

Example Syntax:

[C#] public BitVector32(int data);
[C++] public: BitVector32(int data);
[VB] Public Sub New(ByVal data As Integer)
[JScript] public function BitVector32(data : int); Initializes a new instance of the **System.Collections.Specialized.BitVector32** structure.

Description

Initializes a new instance of the **System.Collections.Specialized.BitVector32** structure containing the data represented in an integer. An integer representing the data of the new **System.Collections.Specialized.BitVector32**.

Properties:

Data

[C#] public int Data {get;}
[C++] public: __property int get_Data();
[VB] Public ReadOnly Property Data As Integer
[JScript] public function get Data() : int;

Description

Gets the value of the **System.Collections.Specialized.BitVector32** as an integer.

To access the value of the individual sections or bit flags, use the **System.Collections.Specialized.BitVector32.Item(System.Int32)** property.

Item

```
[C#] public int this[BitVector32.Section section] {get; set;}
```

```
[C++] public: __property int get_Item(BitVector32.Section section);public:
```

```
__property void set_Item(BitVector32.Section section, int);
```

```
[VB] Public Default Property Item(ByVal section As BitVector32.Section) As
```

Integer

```
[JScript] returnValue =
```

```
BitVector32Object.Item(section);BitVector32Object.Item(section) = returnValue;
```

Description

Gets or sets the value stored in the specified

System.Collections.Specialized.BitVector32.Section .

System.Collections.Specialized.BitVector32.Item(System.Int32)

[Section] property is the indexer for a

System.Collections.Specialized.BitVector32 that is set up as sections, and

System.Collections.Specialized.BitVector32.Item(System.Int32) [int] property

is the indexer for a **System.Collections.Specialized.BitVector32** that is set up as

bit flags. A **System.Collections.Specialized.BitVector32.Section** that contains

the value to get or set.

Item

```
[C#] public bool this[int bit] {get; set;}
```

1 [C++] public: __property bool get_Item(int bit);public: __property void
 2 set_Item(int bit, bool);
 3 [VB] Public Default Property Item(ByVal bit As Integer) As Boolean
 4 [JScript] returnValue = BitVector32Object.Item(bit);BitVector32Object.Item(bit)
 5 = returnValue; Gets or sets the value of the specified section or bit flag.

7 *Description*

8 Gets or sets the state of the bit flag indicated by the specified mask.

9 **System.Collections.Specialized.BitVector32.Item(System.Int32)**

10 [Section] property is the indexer for a
 11 **System.Collections.Specialized.BitVector32** that is set up as sections, and
 12 **System.Collections.Specialized.BitVector32.Item(System.Int32) [int]** property
 13 is the indexer for a **System.Collections.Specialized.BitVector32** that is set up as
 14 bit flags. A mask that indicates the bit to get or set.

15 Methods:

16 CreateMask

17
 18 [C#] public static int CreateMask();
 19 [C++] public: static int CreateMask();
 20 [VB] Public Shared Function CreateMask() As Integer
 21 [JScript] public static function CreateMask() : int; Creates a series of masks that
 22 can be used to access individual bits in a
 23 **System.Collections.Specialized.BitVector32** that is set up as bit flags.

24 *Description*

Creates the first mask in a series of masks that can be used to access individual bits in a **System.Collections.Specialized.BitVector32** that is set up as bit flags.

Return Value: A mask that isolates the first bit flag in the **System.Collections.Specialized.BitVector32**.

Use `CreateMask()` to create the first mask in a series and `CreateMask(int)` for all subsequent masks.

CreateMask

[C#] public static int CreateMask(int previous);

[C++] public: static int CreateMask(int previous);

[VB] Public Shared Function CreateMask(ByVal previous As Integer) As Integer

[JScript] public static function CreateMask(previous : int) : int;

Description

Creates the mask following the specified mask in a series of masks that can be used to access individual bits in a **System.Collections.Specialized.BitVector32** that is set up as bit flags.

Return Value: A mask that isolates the bit flag following the one that *previous* points to in **System.Collections.Specialized.BitVector32**.

Use `CreateMask()` to create the first mask in a series and `CreateMask(int)` for all subsequent masks. The mask that indicates the previous bit flag.

CreateSection

[C#] public static Section CreateSection(short maxVal);

1 [C++] public: static Section CreateSection(short maxValue);

2 [VB] Public Shared Function CreateSection(ByVal maxValue As Short) As

3 Section

4 [JScript] public static function CreateSection(maxValue : Int16) : Section; Creates

5 a series of sections that contain small integers.

6
7 *Description*

8 Creates the first **System.Collections.Specialized.BitVector32.Section** in a
9 series of sections that contain small integers.

10 *Return Value:* A **System.Collections.Specialized.BitVector32.Section** that can
11 hold a number from zero to *maxValue* .

12 A **System.Collections.Specialized.BitVector32.Section** is a window into
13 the **System.Collections.Specialized.BitVector32** and is composed of the smallest
14 number of consecutive bits that can contain the maximum value specified in
15 **System.Collections.Specialized.BitVector32.CreateSection(System.Int16)** . For
16 example, a section with a maximum value of 1 is composed of only one bit,
17 whereas a section with a maximum value of 5 is composed of three bits. You can
18 create a **System.Collections.Specialized.BitVector32.Section** with a maximum
19 value of 1 to serve as a Boolean, thereby allowing you to store integers and
20 Booleans in the same **System.Collections.Specialized.BitVector32** . A 16-bit
21 signed integer that specifies the maximum value for the new
22 **System.Collections.Specialized.BitVector32.Section**.

23 CreateSection

24
25 [C#] public static Section CreateSection(short maxValue, BitVector32.Section

```

1 previous);
2 [C++] public: static Section CreateSection(short maxValue, BitVector32.Section
3 previous);
4 [VB] Public Shared Function CreateSection(ByVal maxValue As Short, ByVal
5 previous As BitVector32.Section) As Section
6 [JScript] public static function CreateSection(maxValue : Int16, previous :
7 BitVector32.Section) : Section;
8

```

Description

Creates a new **System.Collections.Specialized.BitVector32.Section** following the specified **System.Collections.Specialized.BitVector32.Section** in a series of sections that contain small integers.

Return Value: A **System.Collections.Specialized.BitVector32.Section** that can hold a number from zero to *maxValue* .

A **System.Collections.Specialized.BitVector32.Section** is a window into the **System.Collections.Specialized.BitVector32** and is composed of the smallest number of consecutive bits that can contain the maximum value specified in **System.Collections.Specialized.BitVector32.CreateSection(System.Int16)** . For example, a section with a maximum value of 1 is composed of only one bit, whereas a section with a maximum value of 5 is composed of three bits. You can create a **System.Collections.Specialized.BitVector32.Section** with a maximum value of 1 to serve as a Boolean, thereby allowing you to store integers and Booleans in the same **System.Collections.Specialized.BitVector32** . A 16-bit signed integer that specifies the maximum value for the new **System.Collections.Specialized.BitVector32.Section**. The previous

1 **System.Collections.Specialized.BitVector32**.Section in the

2 **System.Collections.Specialized.BitVector32**.

3 Equals

4
5 [C#] public override bool Equals(object o);

6 [C++] public: bool Equals(Object* o);

7 [VB] Overrides Public Function Equals(ByVal o As Object) As Boolean

8 [JScript] public override function Equals(o : Object) : Boolean;

9
10 *Description*

11 Determines whether the specified object is equal to the

12 **System.Collections.Specialized.BitVector32** .

13 *Return Value:* **true** if the specified **System.Object** is equal to the

14 **System.Collections.Specialized.BitVector32** ; otherwise, **false** .

15 The object *o* is considered equal to the

16 **System.Collections.Specialized.BitVector32** if the type of *o* is compatible with

17 the **System.Collections.Specialized.BitVector32** type and if the value of *o* is

18 equal to the value of **System.Collections.Specialized.BitVector32.Data** . The

19 **System.Object** to compare with the current

20 **System.Collections.Specialized.BitVector32**.

21 GetHashCode

22
23 [C#] public override int GetHashCode();

24 [C++] public: int GetHashCode();

25 [VB] Overrides Public Function GetHashCode() As Integer

1 [JScript] public override function GetHashCode() : int;

3 *Description*

4 Serves as a hash function for the

5 **System.Collections.Specialized.BitVector32** .

6 *Return Value:* A hash code for the **System.Collections.Specialized.BitVector32** .

7 The hash code of a **System.Collections.Specialized.BitVector32** is based
8 on the value of **System.Collections.Specialized.BitVector32.Data** . Two
9 instances of **System.Collections.Specialized.BitVector32** with the same value for
10 **System.Collections.Specialized.BitVector32.Data** will also generate the same
11 hash code.

12 ToString

14 [C#] public override string ToString();

15 [C++] public: String* ToString();

16 [VB] Overrides Public Function ToString() As String

17 [JScript] public override function ToString() : String;

19 *Description*

21 ToString

23 [C#] public static string ToString(BitVector32 value);

24 [C++] public: static String* ToString(BitVector32 value);

25 [VB] Public Shared Function ToString(ByVal value As BitVector32) As String

1 [JScript] public static function ToString(value : BitVector32) : String;

2
3 *Description*

4
5 CollectionsUtil class (System.Collections.Specialized)

6 ToString

7
8
9 *Description*

10 Creates collections that ignore the case in strings.

11 These methods generate a case-insensitive instance of the collection using
12 case-insensitive implementations of the hash code provider and the comparer. The
13 resulting instance can be used like any other instances of that class, although it
14 may behave differently.

15 CollectionsUtil

16 *Example Syntax:*

17 ToString

18
19 [C#] public CollectionsUtil();

20 [C++] public: CollectionsUtil();

21 [VB] Public Sub New()

22 [JScript] public function CollectionsUtil();

23 CreateCaseInsensitiveHashtable

24
25 [C#] public static Hashtable CreateCaseInsensitiveHashtable();

1 [C++] public: static Hashtable* CreateCaseInsensitiveHashtable();

2 [VB] Public Shared Function CreateCaseInsensitiveHashtable() As Hashtable

3 [JScript] public static function CreateCaseInsensitiveHashtable() : Hashtable;

4 Creates a new instance of the **System.Collections.Hashtable** class that ignores the
5 case of strings.

6
7 *Description*

8 Creates a new case-insensitive instance of the
9 **System.Collections.Hashtable** class with the default initial capacity.

10 *Return Value:* A new case-insensitive instance of the
11 **System.Collections.Hashtable** class with the default initial capacity.

12 The new **System.Collections.Hashtable** instance uses the default load
13 factor, the **System.Collections.CaseInsensitiveHashCodeProvider** , and the
14 **System.Collections.CaseInsensitiveComparer** .

15 CreateCaseInsensitiveHashtable

16
17 [C#] public static Hashtable CreateCaseInsensitiveHashtable(IDictionary d);

18 [C++] public: static Hashtable* CreateCaseInsensitiveHashtable(IDictionary* d);

19 [VB] Public Shared Function CreateCaseInsensitiveHashtable(ByVal d As
20 IDictionary) As Hashtable

21 [JScript] public static function CreateCaseInsensitiveHashtable(d : IDictionary) :
22 Hashtable;

23
24 *Description*

Copies the entries from the specified dictionary to a new case-insensitive instance of the **System.Collections.Hashtable** class with the same initial capacity as the number of entries copied.

Return Value: A new case-insensitive instance of the **System.Collections.Hashtable** class containing the entries from the specified **System.Collections.IDictionary** .

The new **System.Collections.Hashtable** instance uses the default load factor, the **System.Collections.CaseInsensitiveHashCodeProvider** , and the **System.Collections.CaseInsensitiveComparer** . The **System.Collections.IDictionary** to copy to a new case-insensitive **System.Collections.Hashtable**.

CreateCaseInsensitiveHashtable

[C#] public static Hashtable CreateCaseInsensitiveHashtable(int capacity);

[C++] public: static Hashtable* CreateCaseInsensitiveHashtable(int capacity);

[VB] Public Shared Function CreateCaseInsensitiveHashtable(ByVal capacity As Integer) As Hashtable

[JScript] public static function CreateCaseInsensitiveHashtable(capacity : int) : Hashtable;

Description

Creates a new case-insensitive instance of the **System.Collections.Hashtable** class with the specified initial capacity.

Return Value: A new case-insensitive instance of the

System.Collections.Hashtable class with the specified initial capacity.

The new **System.Collections.Hashtable** instance uses the default load factor, the **System.Collections.CaseInsensitiveHashCodeProvider** , and the **System.Collections.CaseInsensitiveComparer** . The approximate number of entries that the **System.Collections.Hashtable** can initially contain.

CreateCaseInsensitiveSortedList

[C#] public static SortedList CreateCaseInsensitiveSortedList();

[C++] public: static SortedList* CreateCaseInsensitiveSortedList();

[VB] Public Shared Function CreateCaseInsensitiveSortedList() As SortedList

[JScript] public static function CreateCaseInsensitiveSortedList() : SortedList;

Description

Creates a new instance of the **System.Collections.SortedList** class that ignores the case of strings.

Return Value: A new instance of the **System.Collections.SortedList** class that ignores the case of strings.

The new **System.Collections.SortedList** instance is sorted according to the **System.Collections.CaseInsensitiveComparer** .

HybridDictionary class (System.Collections.Specialized)

ToString

Description

1 Implements **IDictionary** by using a
2 **System.Collections.Specialized.ListDictionary** while the collection is small, and
3 then switching to a **System.Collections.Hashtable** when the collection gets large.

4 This class is recommended for cases where the number of elements in a
5 dictionary is unknown. It takes advantage of the improved performance of a
6 **System.Collections.Specialized.ListDictionary** with small collections, and offers
7 the flexibility of switching to a **System.Collections.Hashtable** which handles
8 larger collections better than **System.Collections.Specialized.ListDictionary** .

9 HybridDictionary

10 *Example Syntax:*

11 ToString

12
13 [C#] public HybridDictionary();

14 [C++] public: HybridDictionary();

15 [VB] Public Sub New()

16 [JScript] public function HybridDictionary(); Initializes a new instance of the
17 **System.Collections.Specialized.HybridDictionary** class.

18
19 *Description*

20 Creates an empty case-sensitive
21 **System.Collections.Specialized.HybridDictionary** .

22 By default, the collection is case-sensitive and uses the key's
23 implementation of **System.Object.GetHashCode** as the hash code provider and
24 the key's implementation of **System.Object.Equals(System.Object)** as the
25 comparer.

HybridDictionary

Example Syntax:

ToString

[C#] public HybridDictionary(bool caseInsensitive);

[C++] public: HybridDictionary(bool caseInsensitive);

[VB] Public Sub New(ByVal caseInsensitive As Boolean)

[JScript] public function HybridDictionary(caseInsensitive : Boolean);

Description

Creates an empty **System.Collections.Specialized.HybridDictionary** with the specified case-sensitivity.

If *caseInsensitive* is **false**, the collection uses the key's implementations of **System.Object.GetHashCode** and **System.Object.Equals(System.Object)**. If *caseInsensitive* is **true**, the collection uses the **System.Collections.CaseInsensitiveHashCodeProvider** and a private case-insensitive and culture-insensitive implementation of the **System.Collections.IComparer** interface that only converts the strings to the same case and compares the Unicode values of the characters. A Boolean that denotes whether the **System.Collections.Specialized.HybridDictionary** is case-insensitive.

HybridDictionary

Example Syntax:

ToString

```

1
2 [C#] public HybridDictionary(int initialSize);
3 [C++] public: HybridDictionary(int initialSize);
4 [VB] Public Sub New(ByVal initialSize As Integer)
5 [JScript] public function HybridDictionary(initialSize : int);

```

Description

Creates a case-sensitive

System.Collections.Specialized.HybridDictionary with the specified initial size.

If the initial size of the collection is greater than the optimal size for a **System.Collections.Specialized.ListDictionary**, the collection is stored in a **System.Collections.Hashtable** right away to avoid the overhead of copying elements from the **System.Collections.Specialized.ListDictionary** to the **System.Collections.Hashtable**. The approximate number of entries that the **System.Collections.Specialized.HybridDictionary** can initially contain.

HybridDictionary

Example Syntax:

ToString

```

19
20 [C#] public HybridDictionary(int initialSize, bool caseInsensitive);
21 [C++] public: HybridDictionary(int initialSize, bool caseInsensitive);
22 [VB] Public Sub New(ByVal initialSize As Integer, ByVal caseInsensitive As
23 Boolean)
24 [JScript] public function HybridDictionary(initialSize : int, caseInsensitive :
25 Boolean);

```

Description

Creates a **System.Collections.Specialized.HybridDictionary** with the specified initial size and case-sensitivity.

If the initial size of the collection is greater than the optimal size for a **System.Collections.Specialized.ListDictionary**, the collection is stored in a **System.Collections.Hashtable** right away to avoid the overhead of copying elements from the **System.Collections.Specialized.ListDictionary** to the **System.Collections.Hashtable**. The approximate number of entries that the **System.Collections.Specialized.HybridDictionary** can initially contain. A Boolean that denotes whether the **System.Collections.Specialized.HybridDictionary** is case-insensitive.

Count

ToString

[C#] public int Count {get;}

[C++] public: __property int get_Count();

[VB] Public ReadOnly Property Count As Integer

[JScript] public function get Count() : int;

Description

Gets the number of key-and-value pairs contained in the **System.Collections.Specialized.HybridDictionary**.

IsFixedSize

ToString

1
2 [C#] public bool IsFixedSize {get;}

3 [C++] public: __property bool get_IsFixedSize();

4 [VB] Public ReadOnly Property IsFixedSize As Boolean

5 [JScript] public function get IsFixedSize() : Boolean;

6
7 *Description*

8 Gets a value indicating whether the

9 **System.Collections.Specialized.HybridDictionary** has a fixed size.

10 **System.Collections.Specialized.HybridDictionary** implements the
11 **System.Collections.Specialized.HybridDictionary.IsFixedSize** property because
12 it is required by the interface.

13 IsReadOnly

14 ToString

15
16 [C#] public bool IsReadOnly {get;}

17 [C++] public: __property bool get_IsReadOnly();

18 [VB] Public ReadOnly Property IsReadOnly As Boolean

19 [JScript] public function get IsReadOnly() : Boolean;

20
21 *Description*

22 Gets a value indicating whether the

23 **System.Collections.Specialized.HybridDictionary** is read-only.

System.Collections.Specialized.HybridDictionary implements the
System.Collections.Specialized.HybridDictionary.IsReadOnly property
because it is required by the interface.

IsSynchronized

ToString

[C#] public bool IsSynchronized {get;}

[C++] public: __property bool get_IsSynchronized();

[VB] Public ReadOnly Property IsSynchronized As Boolean

[JScript] public function get IsSynchronized() : Boolean;

Description

Gets a value indicating whether the
System.Collections.Specialized.HybridDictionary is synchronized (thread-safe).

System.Collections.Specialized.HybridDictionary implements the
System.Collections.Specialized.HybridDictionary.IsSynchronized property
because it is required by the interface.

Item

ToString

[C#] public object this[object key] {get; set;}

[C++] public: __property Object* get_Item(Object* key);public: __property void
set_Item(Object* key, Object*);

[VB] Public Default Property Item(ByVal key As Object) As Object

[JScript] returnValue =

HybridDictionaryObject.Item(key);HybridDictionaryObject.Item(key) =
returnValue;

Description

Gets or sets the value associated with the specified key.

This property provides the ability to access a specific element in the collection by using the following syntax: myCollection[key] . The key whose value to get or set.

Keys

ToString

[C#] public ICollection Keys {get;}

[C++] public: __property ICollection* get_Keys();

[VB] Public ReadOnly Property Keys As ICollection

[JScript] public function get Keys() : ICollection;

Description

Gets an **System.Collections.ICollection** containing the keys in the **System.Collections.Specialized.HybridDictionary** .

The order of the values in the **System.Collections.ICollection** is unspecified, but it is the same order as the associated values in the **System.Collections.ICollection** returned by the **System.Collections.Specialized.HybridDictionary.Values** method.

SyncRoot

ToString

```

1
2 [C#] public object SyncRoot {get;}
3 [C++] public: __property Object* get_SyncRoot();
4 [VB] Public ReadOnly Property SyncRoot As Object
5 [JScript] public function get SyncRoot() : Object;
6

```

7 *Description*

8 Gets an object that can be used to synchronize access to the
9 **System.Collections.Specialized.HybridDictionary** .

10 Derived classes can provide their own synchronized version of the
11 **System.Collections.Specialized.HybridDictionary** using the
12 **System.Collections.Specialized.HybridDictionary.SyncRoot** property. The
13 synchronizing code must perform operations on the
14 **System.Collections.Specialized.HybridDictionary.SyncRoot** of the
15 **System.Collections.Specialized.HybridDictionary** , not directly on the
16 **System.Collections.Specialized.HybridDictionary** . This ensures proper
17 operation of collections that are derived from other objects. Specifically, it
18 maintains proper synchronization with other threads that might be simultaneously
19 modifying the **System.Collections.Specialized.HybridDictionary** object.

20 Values

21 ToString

```

22
23 [C#] public ICollection Values {get;}
24 [C++] public: __property ICollection* get_Values();
25 [VB] Public ReadOnly Property Values As ICollection

```

1 [JScript] public function get Values() : ICollection;

3 *Description*

4 Gets an **System.Collections.ICollection** containing the values in the
5 **System.Collections.Specialized.HybridDictionary** .

6 The order of the values in the **System.Collections.ICollection** is
7 unspecified, but it is the same order as the associated keys in the
8 **System.Collections.ICollection** returned by the
9 **System.Collections.Specialized.HybridDictionary.Keys** method.

10 Add

12 [C#] public void Add(object key, object value);

13 [C++] public: __sealed void Add(Object* key, Object* value);

14 [VB] NotOverridable Public Sub Add(ByVal key As Object, ByVal value As
15 Object)

16 [JScript] public function Add(key : Object, value : Object);

18 *Description*

19 Adds an entry with the specified key and value into the
20 **System.Collections.Specialized.HybridDictionary** .

21 An object that has no correlation between its state and its hash code value
22 should typically not be used as the key. For example, String objects are better than
23 StringBuilder objects for use as keys. The key of the entry to add. The value of the
24 entry to add.

25 Clear

[C#] public void Clear();

[C++] public: __sealed void Clear();

[VB] NotOverridable Public Sub Clear()

[JScript] public function Clear();

Description

Removes all entries from the

System.Collections.Specialized.HybridDictionary .

System.Collections.Specialized.HybridDictionary.Count is set to zero.

Contains

[C#] public bool Contains(object key);

[C++] public: __sealed bool Contains(Object* key);

[VB] NotOverridable Public Function Contains(ByVal key As Object) As Boolean

[JScript] public function Contains(key : Object) : Boolean;

Description

Determines whether the

System.Collections.Specialized.HybridDictionary contains a specific key.

Return Value: **true** if the **System.Collections.Specialized.HybridDictionary** contains an entry with the specified key; otherwise, **false** .

This implementation is close to O(1) in most cases. The key to locate in the

System.Collections.Specialized.HybridDictionary.

CopyTo

1
2 [C#] public void CopyTo(Array array, int index);

3 [C++] public: __sealed void CopyTo(Array* array, int index);

4 [VB] NotOverridable Public Sub CopyTo(ByVal array As Array, ByVal index As
5 Integer)

6 [JScript] public function CopyTo(array : Array, index : int);

7
8 *Description*

9 Copies the **System.Collections.Specialized.HybridDictionary** entries to a
10 one-dimensional **System.Array** instance at the specified index.

11 The elements are copied to the **System.Array** in the same order in which
12 the enumerator iterates through the

13 **System.Collections.Specialized.HybridDictionary** . The one-dimensional
14 **System.Array** that is the destination of the **System.Collections.DictionaryEntry**
15 objects copied from **System.Collections.Specialized.HybridDictionary**. The
16 **System.Array** must have zero-based indexing. The zero-based index in *array* at
17 which copying begins.

18 GetEnumerator

19
20 [C#] public IDictionaryEnumerator GetEnumerator();

21 [C++] public: __sealed IDictionaryEnumerator* GetEnumerator();

22 [VB] NotOverridable Public Function GetEnumerator() As IDictionaryEnumerator

23 [JScript] public function GetEnumerator() : IDictionaryEnumerator; Returns an
24 enumerator that can iterate through the

25 **System.Collections.Specialized.HybridDictionary** .

Description

Returns an enumerator that can iterate through the

System.Collections.Specialized.HybridDictionary .

Return Value: An **System.Collections.IDictionaryEnumerator** for the

System.Collections.Specialized.HybridDictionary .

Enumerators are intended to be used only to read data in the collection.

Enumerators cannot be used to modify the underlying collection.

Remove

[C#] public void Remove(object key);

[C++] public: __sealed void Remove(Object* key);

[VB] NotOverridable Public Sub Remove(ByVal key As Object)

[JScript] public function Remove(key : Object);

Description

Removes the entry with the specified key from the

System.Collections.Specialized.HybridDictionary .

If the **System.Collections.Specialized.HybridDictionary** does not contain an element with the specified key, the

System.Collections.Specialized.HybridDictionary remains unchanged. No exception is thrown. The key of the entry to remove.

IEnumerator.GetEnumerator

[C#] IEnumerator IEnumerable.GetEnumerator();

```

1 [C++] IEnumerator* IEnumerable::GetEnumerator();
2 [VB] Function GetEnumerator() As IEnumerator Implements
3 IEnumerable.GetEnumerator
4 [JScript] function IEnumerable.GetEnumerator() : IEnumerator;
5     NameObjectCollectionBase.KeysCollection class
6 (System.Collections.Specialized)
7     ToString
8
9

```

Description

Represents a collection of the **System.String** keys of a collection.

Count

ToString

```

15 [C#] public int Count {get;}
16 [C++] public: __property int get_Count();
17 [VB] Public ReadOnly Property Count As Integer
18 [JScript] public function get Count() : int;
19

```

Description

Gets the number of keys in the

System.Collections.Specialized.NameObjectCollectionBase.KeysCollection .

Item

ToString

1 [C#] public string this[int index] {get;}

2 [C++] public: __property String* get_Item(int index);

3 [VB] Public Default ReadOnly Property Item(ByVal index As Integer) As String

4 [JScript] returnValue = KeysCollectionObject.Item(index);

5 6 7 *Description*

8 Gets the entry at the specified index of the collection.

9 This property provides the ability to access a specific element in the
10 collection by using the following syntax: myCollection[index] (In Visual Basic,
11 myCollection(index)). The zero-based index of the entry to locate in the
12 collection.

13 *Get*

14
15 [C#] public virtual string Get(int index);

16 [C++] public: virtual String* Get(int index);

17 [VB] Overridable Public Function Get(ByVal index As Integer) As String

18 [JScript] public function Get(index : int) : String;

19 20 *Description*

21 Gets the key at the specified index of the collection.

22 *Return Value:* A **System.String** that contains the key at the specified index of the
23 collection. The zero-based index of the key to get from the collection.

24 *GetEnumerator*

1
2 [C#] public IEnumerator GetEnumerator();

3 [C++] public: __sealed IEnumerator* GetEnumerator();

4 [VB] NotOverridable Public Function GetEnumerator() As IEnumerator

5 [JScript] public function GetEnumerator() : IEnumerator;

6
7 *Description*

8 Returns an enumerator that can iterate through the

9 **System.Collections.Specialized.NameObjectCollectionBase.KeysCollection** .

10 *Return Value:* An **System.Collections.IEnumerator** for the

11 **System.Collections.Specialized.NameObjectCollectionBase.KeysCollection** .

12 This enumerator returns the keys of the collection as strings.

13 **ICollection.CopyTo**

14
15 [C#] void ICollection.CopyTo(Array array, int index);

16 [C++] void ICollection::CopyTo(Array* array, int index);

17 [VB] Sub CopyTo(ByVal array As Array, ByVal index As Integer) Implements

18 **ICollection.CopyTo**

19 [JScript] function ICollection.CopyTo(array : Array, index : int);

20 **ListDictionary** class (**System.Collections.Specialized**)

21 **ToString**

22
23
24 *Description*

1 Implements **IDictionary** using a singly linked list for collections that
2 contain 10 items or less.

3 This is a simple implementation of **System.Collections.IDictionary** using
4 a singly linked list. It is smaller and faster than a **System.Collections.Hashtable** if
5 the number of elements is 10 or less. This should not be used if performance is
6 important for large numbers of elements.

7 ListDictionary

8 *Example Syntax:*

9 ToString

10
11 [C#] public ListDictionary();

12 [C++] public: ListDictionary();

13 [VB] Public Sub New()

14 [JScript] public function ListDictionary(); Initializes a new instance of the
15 **System.Collections.Specialized.ListDictionary** class.

16
17 *Description*

18 Creates an empty **System.Collections.Specialized.ListDictionary** using
19 the default comparer.

20 The comparer determines whether two keys are equal. Every key in a
21 **System.Collections.Specialized.ListDictionary** must be unique. The default
22 comparer is the key's implementation of **System.Object.Equals(System.Object)** .

23 ListDictionary

24 *Example Syntax:*

25 ToString

```

1
2 [C#] public ListDictionary(IComparer comparer);
3 [C++] public: ListDictionary(IComparer* comparer);
4 [VB] Public Sub New(ByVal comparer As IComparer)
5 [JScript] public function ListDictionary(comparer : IComparer);
6

```

Description

Creates an empty **System.Collections.Specialized.ListDictionary** using the specified comparer.

The comparer determines whether two keys are equal. Every key in a **System.Collections.Specialized.ListDictionary** must be unique. The default comparer is the key's implementation of **System.Object.Equals(System.Object)** . The **System.Collections.IComparer** to use to determine whether two keys are equal.

Count

ToString

```

17
18 [C#] public int Count {get;}
19 [C++] public: __property int get_Count();
20 [VB] Public ReadOnly Property Count As Integer
21 [JScript] public function get Count() : int;
22

```

Description

Gets the number of key-and-value pairs contained in the **System.Collections.Specialized.ListDictionary** .

IsFixedSize

ToString

[C#] public bool IsFixedSize {get;}

[C++] public: __property bool get_IsFixedSize();

[VB] Public ReadOnly Property IsFixedSize As Boolean

[JScript] public function get IsFixedSize() : Boolean;

Description

Gets a value indicating whether the

System.Collections.Specialized.ListDictionary has a fixed size.

System.Collections.Specialized.ListDictionary implements the

System.Collections.Specialized.ListDictionary.IsFixedSize property because it is required by the interface.

IsReadOnly

ToString

[C#] public bool IsReadOnly {get;}

[C++] public: __property bool get_IsReadOnly();

[VB] Public ReadOnly Property IsReadOnly As Boolean

[JScript] public function get IsReadOnly() : Boolean;

Description

Gets a value indicating whether the

System.Collections.Specialized.ListDictionary is read-only.

System.Collections.Specialized.ListDictionary implements the **System.Collections.Specialized.ListDictionary.IsReadOnly** property because it is required by the interface.

IsSynchronized

ToString

[C#] public bool IsSynchronized {get;}

[C++] public: __property bool get_IsSynchronized();

[VB] Public ReadOnly Property IsSynchronized As Boolean

[JScript] public function get IsSynchronized() : Boolean;

Description

Gets a value indicating whether the **System.Collections.Specialized.ListDictionary** is synchronized (thread-safe).

System.Collections.Specialized.ListDictionary implements the **System.Collections.Specialized.ListDictionary.IsSynchronized** property because it is required by the interface.

Item

ToString

[C#] public object this[object key] {get; set;}

[C++] public: __property Object* get_Item(Object* key);public: __property void set_Item(Object* key, Object*);

[VB] Public Default Property Item(ByVal key As Object) As Object

[JScript] returnValue =

1 ListDictionaryObject.Item(key);ListDictionaryObject.Item(key) = returnValue;

2
3 *Description*

4 Gets or sets the value associated with the specified key.

5 This property provides the ability to access a specific element in the
6 collection by using the following syntax: myCollection[key] . The key whose
7 value to get or set.

8 Keys

9 ToString

10
11 [C#] public ICollection Keys {get;}

12 [C++] public: __property ICollection* get_Keys();

13 [VB] Public ReadOnly Property Keys As ICollection

14 [JScript] public function get Keys() : ICollection;

15
16 *Description*

17 Gets an **System.Collections.ICollection** containing the keys in the
18 **System.Collections.Specialized.ListDictionary** .

19 The order of the values in the **System.Collections.ICollection** is
20 unspecified, but it is the same order as the associated values in the
21 **System.Collections.ICollection** returned by the
22 **System.Collections.Specialized.ListDictionary.Values** method.

23 SyncRoot

24 ToString

1
2 [C#] public object SyncRoot {get;}
3 [C++] public: __property Object* get_SyncRoot();
4 [VB] Public ReadOnly Property SyncRoot As Object
5 [JScript] public function get SyncRoot() : Object;
6

7 *Description*

8 Gets an object that can be used to synchronize access to the

9 **System.Collections.Specialized.ListDictionary** .

10 Derived classes can provide their own synchronized version of the

11 **System.Collections.Specialized.ListDictionary** using the

12 **System.Collections.Specialized.ListDictionary.SyncRoot** property. The

13 synchronizing code must perform operations on the

14 **System.Collections.Specialized.ListDictionary.SyncRoot** of the

15 **System.Collections.Specialized.ListDictionary** , not directly on the

16 **System.Collections.Specialized.ListDictionary** . This ensures proper operation

17 of collections that are derived from other objects. Specifically, it maintains proper

18 synchronization with other threads that might be simultaneously modifying the

19 **System.Collections.Specialized.ListDictionary** object.

20 Values

21 ToString

22
23 [C#] public ICollection Values {get;}
24 [C++] public: __property ICollection* get_Values();
25 [VB] Public ReadOnly Property Values As ICollection

1 [JScript] public function get Values() : ICollection;

3 *Description*

4 Gets an **System.Collections.ICollection** containing the values in the
5 **System.Collections.Specialized.ListDictionary** .

6 The order of the values in the **System.Collections.ICollection** is
7 unspecified, but it is the same order as the associated keys in the
8 **System.Collections.ICollection** returned by the
9 **System.Collections.Specialized.ListDictionary.Keys** method.

10 Add

12 [C#] public void Add(object key, object value);

13 [C++] public: __sealed void Add(Object* key, Object* value);

14 [VB] NotOverridable Public Sub Add(ByVal key As Object, ByVal value As
15 Object)

16 [JScript] public function Add(key : Object, value : Object);

18 *Description*

19 Adds an entry with the specified key and value into the
20 **System.Collections.Specialized.ListDictionary** .

21 An object that has no correlation between its state and its hash code value
22 should typically not be used as the key. For example, String objects are better than
23 StringBuilder objects for use as keys. The key of the entry to add. The value of the
24 entry to add.

25 Clear

1
2 [C#] public void Clear();

3 [C++] public: __sealed void Clear();

4 [VB] NotOverridable Public Sub Clear()

5 [JScript] public function Clear();

6
7 *Description*

8 Removes all entries from the

9 **System.Collections.Specialized.ListDictionary** .

10 **System.Collections.Specialized.ListDictionary.Count** is set to zero.

11 Contains

12
13 [C#] public bool Contains(object key);

14 [C++] public: __sealed bool Contains(Object* key);

15 [VB] NotOverridable Public Function Contains(ByVal key As Object) As Boolean

16 [JScript] public function Contains(key : Object) : Boolean;

17
18 *Description*

19 Determines whether the **System.Collections.Specialized.ListDictionary**
20 contains a specific key.

21 *Return Value:* **true** if the **System.Collections.Specialized.ListDictionary**
22 contains an entry with the specified key; otherwise, **false** .

23 This implementation is close to O(1) in most cases. The key to locate in the
24 **System.Collections.Specialized.ListDictionary**.

25 CopyTo

1
2 [C#] public void CopyTo(Array array, int index);

3 [C++] public: __sealed void CopyTo(Array* array, int index);

4 [VB] NotOverridable Public Sub CopyTo(ByVal array As Array, ByVal index As
5 Integer)

6 [JScript] public function CopyTo(array : Array, index : int);

7
8 *Description*

9 Copies the **System.Collections.Specialized.ListDictionary** entries to a
10 one-dimensional **System.Array** instance at the specified index.

11 The elements are copied to the **System.Array** in the same order in which
12 the enumerator iterates through the

13 **System.Collections.Specialized.ListDictionary**. The one-dimensional
14 **System.Array** that is the destination of the **System.Collections.DictionaryEntry**
15 objects copied from **System.Collections.Specialized.ListDictionary**. The
16 **System.Array** must have zero-based indexing. The zero-based index in *array* at
17 which copying begins.

18 GetEnumerator

19
20 [C#] public IDictionaryEnumerator GetEnumerator();

21 [C++] public: __sealed IDictionaryEnumerator* GetEnumerator();

22 [VB] NotOverridable Public Function GetEnumerator() As IDictionaryEnumerator

23 [JScript] public function GetEnumerator() : IDictionaryEnumerator;

24
25 *Description*

1 Returns an enumerator that can iterate through the
2 **System.Collections.Specialized.ListDictionary** .

3 *Return Value:* An **System.Collections.IDictionaryEnumerator** for the
4 **System.Collections.Specialized.ListDictionary** .

5 Enumerators are intended to be used only to read data in the collection.
6 Enumerators cannot be used to modify the underlying collection.

7 Remove

8
9 [C#] public void Remove(object key);

10 [C++] public: __sealed void Remove(Object* key);

11 [VB] NotOverridable Public Sub Remove(ByVal key As Object)

12 [JScript] public function Remove(key : Object);

13 14 *Description*

15 Removes the entry with the specified key from the
16 **System.Collections.Specialized.ListDictionary** .

17 If the **System.Collections.Specialized.ListDictionary** does not contain an
18 element with the specified key, the
19 **System.Collections.Specialized.ListDictionary** remains unchanged. No
20 exception is thrown. The key of the entry to remove.

21 IEnumerable.GetEnumerator

22
23 [C#] IEnumerator IEnumerable.GetEnumerator();

24 [C++] IEnumerator* IEnumerable::GetEnumerator();

25 [VB] Function GetEnumerator() As IEnumerator Implements

1 IEnumerable.GetEnumerator

2 [JScript] function IEnumerable.GetEnumerator() : IEnumerator;

3 NameObjectCollectionBase class (System.Collections.Specialized)

4 ToString

7 *Description*

8 Provides the **abstract** base class for a sorted collection of associated
9 **System.String** keys and **System.Object** values that can be accessed either with
10 the key or with the index.

11 The underlying structure for this class is a hashtable.

12 NameObjectCollectionBase

13 *Example Syntax:*

14 ToString

16 [C#] protected NameObjectCollectionBase();

17 [C++] protected: NameObjectCollectionBase();

18 [VB] Protected Sub New()

19 [JScript] protected function NameObjectCollectionBase(); Initializes a new
20 instance of the **System.Collections.Specialized.NameObjectCollectionBase**
21 class.

23 *Description*

24 Initializes a new instance of the
25 **System.Collections.Specialized.NameObjectCollectionBase** class that is empty.

The capacity is the number of key-and-value pairs that the **System.Collections.Specialized.NameObjectCollectionBase** instance can contain. The default initial capacity is zero. The capacity is automatically increased as required.

NameObjectCollectionBase

Example Syntax:

ToString

[C#] protected **NameObjectCollectionBase**(int capacity);

[C++] protected: **NameObjectCollectionBase**(int capacity);

[VB] Protected Sub New(ByVal capacity As Integer)

[JScript] protected function **NameObjectCollectionBase**(capacity : int);

Description

Initializes a new instance of the **System.Collections.Specialized.NameObjectCollectionBase** class that is empty and has the specified initial capacity.

The capacity is the number of key-and-value pairs that the **System.Collections.Specialized.NameObjectCollectionBase** instance can contain. Specifying the initial capacity eliminates the need to perform a number of resizing operations while entries are added to the **System.Collections.Specialized.NameObjectCollectionBase** instance. The capacity is automatically increased as required. The approximate number of entries that the **System.Collections.Specialized.NameObjectCollectionBase** instance can initially contain.

NameObjectCollectionBase

Example Syntax:

ToString

[C#] protected NameObjectCollectionBase(IHashCodeProvider hashProvider,
IComparer comparer);

[C++] protected: NameObjectCollectionBase(IHashCodeProvider* hashProvider,
IComparer* comparer);

[VB] Protected Sub New(ByVal hashProvider As IHashCodeProvider, ByVal
comparer As IComparer)

[JScript] protected function NameObjectCollectionBase(hashProvider :
IHashCodeProvider, comparer : IComparer);

Description

Initializes a new instance of the
System.Collections.Specialized.NameObjectCollectionBase class that is empty
and uses the specified hash code provider and the specified comparer.

The capacity is the number of key-and-value pairs that the
System.Collections.Specialized.NameObjectCollectionBase instance can
contain. The default initial capacity is zero. The capacity is automatically
increased as required. The **System.Collections.IHashCodeProvider** that will
supply the hash codes for all keys in the
System.Collections.Specialized.NameObjectCollectionBase instance. The
System.Collections.IComparer to use to determine whether two keys are equal.

NameObjectCollectionBase

Example Syntax:

ToString

[C#] protected NameObjectCollectionBase(SerializationInfo info,
StreamingContext context);

[C++] protected: NameObjectCollectionBase(SerializationInfo* info,
StreamingContext context);

[VB] Protected Sub New(ByVal info As SerializationInfo, ByVal context As
StreamingContext)

[JScript] protected function NameObjectCollectionBase(info : SerializationInfo,
context : StreamingContext);

Description

Initializes a new instance of the
System.Collections.Specialized.NameObjectCollectionBase class that is
serializable and uses the specified
System.Runtime.Serialization.SerializationInfo and
System.Runtime.Serialization.StreamingContext . A
System.Runtime.Serialization.SerializationInfo object that contains the
information required to serialize the new
System.Collections.Specialized.NameObjectCollectionBase instance. A
System.Runtime.Serialization.StreamingContext object that contains the source
and destination of the serialized stream associated with the new
System.Collections.Specialized.NameObjectCollectionBase instance.

NameObjectCollectionBase

Example Syntax:

ToString

[C#] protected NameObjectCollectionBase(int capacity, IHashCodeProvider hashProvider, IComparer comparer);

[C++] protected: NameObjectCollectionBase(int capacity, IHashCodeProvider* hashProvider, IComparer* comparer);

[VB] Protected Sub New(ByVal capacity As Integer, ByVal hashProvider As IHashCodeProvider, ByVal comparer As IComparer)

[JScript] protected function NameObjectCollectionBase(capacity : int, hashProvider : IHashCodeProvider, comparer : IComparer);

Description

Initializes a new instance of the **System.Collections.Specialized.NameObjectCollectionBase** class that is empty, has the specified initial capacity and uses the specified case-insensitive hash code provider and the specified case-insensitive comparer.

The capacity is the number of key-and-value pairs that the **System.Collections.Specialized.NameObjectCollectionBase** instance can contain. Specifying the initial capacity eliminates the need to perform a number of resizing operations while entries are added to the **System.Collections.Specialized.NameObjectCollectionBase** instance. The capacity is automatically increased as required. The approximate number of entries that the **System.Collections.Specialized.NameObjectCollectionBase** instance can initially contain. The case-insensitive

1 **System.Collections.IHashCodeProvider** that will supply the hash codes for all
2 keys in the **System.Collections.Specialized.NameObjectCollectionBase**
3 instance. The case-insensitive **System.Collections.IComparer** to use to determine
4 whether two keys are equal.

5 Count

6 ToString

7
8 [C#] public virtual int Count {get;}

9 [C++] public: __property virtual int get_Count();

10 [VB] Overridable Public ReadOnly Property Count As Integer

11 [JScript] public function get Count() : int;

12
13 *Description*

14 Gets the number of key-and-value pairs contained in the
15 **System.Collections.Specialized.NameObjectCollectionBase** instance.

16 IsReadOnly

17 ToString

18
19 [C#] protected bool IsReadOnly {get; set;}

20 [C++] protected: __property bool get_IsReadOnly();protected: __property void
21 set_IsReadOnly(bool);

22 [VB] Protected Property IsReadOnly As Boolean

23 [JScript] protected function get IsReadOnly() : Boolean;protected function set
24 IsReadOnly(Boolean);

Description

Gets or sets a value indicating whether the **System.Collections.Specialized.NameObjectCollectionBase** instance is read-only.

Keys

ToString

[C#] public virtual NameObjectCollectionBase.KeysCollection Keys {get;}

[C++] public: __property virtual NameObjectCollectionBase.KeysCollection* get_Keys();

[VB] Overridable Public ReadOnly Property Keys As

NameObjectCollectionBase.KeysCollection

[JScript] public function get Keys() : NameObjectCollectionBase.KeysCollection;

Description

Gets a **System.Collections.Specialized.NameObjectCollectionBase.KeysCollection** instance that contains all the keys in the **System.Collections.Specialized.NameObjectCollectionBase** instance.

BaseAdd

[C#] protected void BaseAdd(string name, object value);

[C++] protected: void BaseAdd(String* name, Object* value);

[VB] Protected Sub BaseAdd(ByVal name As String, ByVal value As Object)

1 [JScript] protected function BaseAdd(name : String, value : Object);

3 *Description*

4 Adds an entry with the specified key and value into the
5 **System.Collections.Specialized.NameObjectCollectionBase** instance. The
6 **System.String** key of the entry to add. The key can be **null**. The **System.Object**
7 value of the entry to add. The value can be **null**.

8 BaseClear

10 [C#] protected void BaseClear();

11 [C++] protected: void BaseClear();

12 [VB] Protected Sub BaseClear()

13 [JScript] protected function BaseClear();

15 *Description*

16 Removes all entries from the
17 **System.Collections.Specialized.NameObjectCollectionBase** instance.
18 **System.Collections.Specialized.NameObjectCollectionBase.Count** is set
19 to zero.

20 BaseGet

22 [C#] protected object BaseGet(int index);

23 [C++] protected: Object* BaseGet(int index);

24 [VB] Protected Function BaseGet(ByVal index As Integer) As Object

25 [JScript] protected function BaseGet(index : int) : Object;

Description

Gets the value of the entry at the specified index of the **System.Collections.Specialized.NameObjectCollectionBase** instance.

Return Value: An **System.Object** that represents the value of the entry at the specified index. The zero-based index of the value to get.

BaseGet

[C#] protected object BaseGet(string name);

[C++] protected: Object* BaseGet(String* name);

[VB] Protected Function BaseGet(ByVal name As String) As Object

[JScript] protected function BaseGet(name : String) : Object; Gets the value of the specified entry from the

System.Collections.Specialized.NameObjectCollectionBase instance.

Description

Gets the value of the first entry with the specified key from the **System.Collections.Specialized.NameObjectCollectionBase** instance.

Return Value: An **System.Object** that represents the value of the first entry with the specified key, if found; otherwise, **null**.

If the collection contains multiple entries with the specified key, this method returns only the first entry. To get the values of subsequent entries with the same key, use the enumerator to iterate through the collection and compare the keys. The **System.String** key of the entry to get. The key can be **null**.

BaseGetAllKeys

[C#] protected string[] BaseGetAllKeys();

[C++] protected: String* BaseGetAllKeys() __gc[];

[VB] Protected Function BaseGetAllKeys() As String()

[JScript] protected function BaseGetAllKeys() : String[];

Description

Returns a **System.String** array that contains all the keys in the **System.Collections.Specialized.NameObjectCollectionBase** instance.

Return Value: A **System.String** array that contains all the keys in the **System.Collections.Specialized.NameObjectCollectionBase** instance.

BaseGetAllValues

[C#] protected object[] BaseGetAllValues();

[C++] protected: Object* BaseGetAllValues() __gc[];

[VB] Protected Function BaseGetAllValues() As Object()

[JScript] protected function BaseGetAllValues() : Object[]; Returns an array that contains all the values in the

System.Collections.Specialized.NameObjectCollectionBase instance.

Description

Returns an **System.Object** array that contains all the values in the **System.Collections.Specialized.NameObjectCollectionBase** instance.

Return Value: An **System.Object** array that contains all the values in the **System.Collections.Specialized.NameObjectCollectionBase** instance.

BaseGetAllValues

[C#] protected object[] BaseGetAllValues(Type type);
[C++] protected: Object* BaseGetAllValues(Type* type) __gc[];
[VB] Protected Function BaseGetAllValues(ByVal type As Type) As Object()
[JScript] protected function BaseGetAllValues(type : Type) : Object[];

Description

Returns an array of the specified type that contains all the values in the **System.Collections.Specialized.NameObjectCollectionBase** instance.

Return Value: An array of the specified type that contains all the values in the **System.Collections.Specialized.NameObjectCollectionBase** instance. A **System.Type** that represents the type of array to return.

BaseGetKey

[C#] protected string BaseGetKey(int index);
[C++] protected: String* BaseGetKey(int index);
[VB] Protected Function BaseGetKey(ByVal index As Integer) As String
[JScript] protected function BaseGetKey(index : int) : String;

Description

Gets the key of the entry at the specified index of the **System.Collections.Specialized.NameObjectCollectionBase** instance.
Return Value: A **System.String** that represents the key of the entry at the specified index. The zero-based index of the key to get.

BaseHasKeys

[C#] protected bool BaseHasKeys();

[C++] protected: bool BaseHasKeys();

[VB] Protected Function BaseHasKeys() As Boolean

[JScript] protected function BaseHasKeys() : Boolean;

Description

Gets a value indicating whether the **System.Collections.Specialized.NameObjectCollectionBase** instance contains entries whose keys are not **null**.

Return Value: **true** if the

System.Collections.Specialized.NameObjectCollectionBase instance contains entries whose keys are not **null**; otherwise, **false**.

BaseRemove

[C#] protected void BaseRemove(string name);

[C++] protected: void BaseRemove(String* name);

[VB] Protected Sub BaseRemove(ByVal name As String)

[JScript] protected function BaseRemove(name : String); Removes the specified entries from the **System.Collections.Specialized.NameObjectCollectionBase** instance.

Description

1 Removes the entries with the specified key from the
2 **System.Collections.Specialized.NameObjectCollectionBase** instance.

3 In collections such as lists, queues and stacks, the elements that follow the
4 removed element move up to occupy the vacated spot. The **System.String** key of
5 the entries to remove. The key can be **null**.

6 BaseRemoveAt

7
8 [C#] protected void BaseRemoveAt(int index);

9 [C++] protected: void BaseRemoveAt(int index);

10 [VB] Protected Sub BaseRemoveAt(ByVal index As Integer)

11 [JScript] protected function BaseRemoveAt(index : int);

12 13 *Description*

14 Removes the entry at the specified index of the
15 **System.Collections.Specialized.NameObjectCollectionBase** instance.

16 In collections such as lists, queues and stacks, the elements that follow the
17 removed element move up to occupy the vacated spot. The zero-based index of the
18 entry to remove.

19 BaseSet

20
21 [C#] protected void BaseSet(int index, object value);

22 [C++] protected: void BaseSet(int index, Object* value);

23 [VB] Protected Sub BaseSet(ByVal index As Integer, ByVal value As Object)

24 [JScript] protected function BaseSet(index : int, value : Object);

1
2 *Description*

3 Sets the value of the entry at the specified index of the
4 **System.Collections.Specialized.NameObjectCollectionBase** instance. The zero-
5 based index of the entry to set. The **System.Object** that represents the new value
6 of the entry to set. The value can be **null**.

7 BaseSet

8
9 [C#] protected void BaseSet(string name, object value);
10 [C++] protected: void BaseSet(String* name, Object* value);
11 [VB] Protected Sub BaseSet(ByVal name As String, ByVal value As Object)
12 [JScript] protected function BaseSet(name : String, value : Object); Sets the value
13 of an entry in the **System.Collections.Specialized.NameObjectCollectionBase**
14 instance.
15

16 *Description*

17 Sets the value of the first entry with the specified key in the
18 **System.Collections.Specialized.NameObjectCollectionBase** instance, if found;
19 otherwise, adds an entry with the specified key and value into the
20 **System.Collections.Specialized.NameObjectCollectionBase** instance.

21 If the collection contains multiple entries with the specified key, this
22 method sets only the first entry. To set the values of subsequent entries with the
23 same key, use the enumerator to iterate through the collection and compare the
24 keys. The **System.String** key of the entry to set. The key can be **null**. The
25

System.Object that represents the new value of the entry to set. The value can be **null**.

GetEnumerator

[C#] public IEnumerator GetEnumerator();

[C++] public: __sealed IEnumerator* GetEnumerator();

[VB] NotOverridable Public Function GetEnumerator() As IEnumerator

[JScript] public function GetEnumerator() : IEnumerator;

Description

Returns an enumerator that can iterate through the

System.Collections.Specialized.NameObjectCollectionBase .

Return Value: An **System.Collections.IEnumerator** for the

System.Collections.Specialized.NameObjectCollectionBase instance.

This enumerator returns the keys of the collection as strings.

GetObjectData

[C#] public virtual void GetObjectData(SerializationInfo info, StreamingContext context);

[C++] public: virtual void GetObjectData(SerializationInfo* info, StreamingContext context);

[VB] Overridable Public Sub GetObjectData(ByVal info As SerializationInfo, ByVal context As StreamingContext)

[JScript] public function GetObjectData(info : SerializationInfo, context : StreamingContext);

Description

Implements the **System.Runtime.Serialization.ISerializable** interface and returns the data needed to serialize the **System.Collections.Specialized.NameObjectCollectionBase** instance. A **System.Runtime.Serialization.SerializationInfo** object that contains the information required to serialize the **System.Collections.Specialized.NameObjectCollectionBase** instance. A **System.Runtime.Serialization.StreamingContext** object that contains the source and destination of the serialized stream associated with the **System.Collections.Specialized.NameObjectCollectionBase** instance.

OnDeserialization

[C#] public virtual void OnDeserialization(object sender);
[C++] public: virtual void OnDeserialization(Object* sender);
[VB] Overridable Public Sub OnDeserialization(ByVal sender As Object)
[JScript] public function OnDeserialization(sender : Object);

Description

Implements the **System.Runtime.Serialization.ISerializable** interface and raises the deserialization event when the deserialization is complete. The source of the deserialization event.

ICollection.CopyTo

[C#] void ICollection.CopyTo(Array array, int index);

1 [C++] void ICollection::CopyTo(Array* array, int index);

2 [VB] Sub CopyTo(ByVal array As Array, ByVal index As Integer) Implements

3 ICollection.CopyTo

4 [JScript] function ICollection.CopyTo(array : Array, index : int);

5 NameValueCollection class (System.Collections.Specialized)

6 ToString

7
8
9 *Description*

10 Represents a sorted collection of associated **System.String** keys and
11 **System.String** values that can be accessed either with the key or with the index.

12 This collection is based on the
13 **System.Collections.Specialized.NameObjectCollectionBase** class. However,
14 unlike the **System.Collections.Specialized.NameObjectCollectionBase** , this
15 class stores multiple string values under a single key.

16 NameValueCollection

17 *Example Syntax:*

18 ToString

19
20 [C#] public NameValueCollection();

21 [C++] public: NameValueCollection();

22 [VB] Public Sub New()

23 [JScript] public function NameValueCollection(); Initializes a new instance of the

24 **System.Collections.Specialized.NameValueCollection** class.

25

1
2 *Description*

3 Initializes a new instance of the
4 **System.Collections.Specialized.NameValueCollection** class that is empty, has
5 the default initial capacity and uses the default case-insensitive hash code provider
6 and the default case-insensitive comparer.

7 The capacity is the number of key-and-value pairs that the
8 **System.Collections.Specialized.NameValueCollection** can contain. The default
9 initial capacity is zero. The capacity is automatically increased as required.

10 NameValueCollection

11 *Example Syntax:*

12 ToString

13
14 [C#] public NameValueCollection(int capacity);

15 [C++] public: NameValueCollection(int capacity);

16 [VB] Public Sub New(ByVal capacity As Integer)

17 [JScript] public function NameValueCollection(capacity : int);

18
19 *Description*

20 Initializes a new instance of the
21 **System.Collections.Specialized.NameValueCollection** class that is empty, has
22 the specified initial capacity and uses the default case-insensitive hash code
23 provider and the default case-insensitive comparer.

24 The capacity is the number of key-and-value pairs that the
25 **System.Collections.Specialized.NameValueCollection** can contain. The default

initial capacity is zero. The capacity is automatically increased as required. The initial number of entries that the

System.Collections.Specialized.NameValueCollection can contain.

NameValueCollection

Example Syntax:

ToString

[C#] public NameValueCollection(NameValueCollection col);

[C++] public: NameValueCollection(NameValueCollection* col);

[VB] Public Sub New(ByVal col As NameValueCollection)

[JScript] public function NameValueCollection(col : NameValueCollection);

Description

Copies the entries from the specified

System.Collections.Specialized.NameValueCollection to a new

System.Collections.Specialized.NameValueCollection with the same initial capacity as the number of entries copied and using the same hash code provider and the same comparer as the source collection.

The capacity is the number of key-and-value pairs that the

System.Collections.Specialized.NameValueCollection can contain. The default initial capacity is zero. The capacity is automatically increased as required. The

System.Collections.Specialized.NameValueCollection to copy to the new

System.Collections.Specialized.NameValueCollection instance.

NameValueCollection

Example Syntax:

ToString

[C#] public NameValueCollection(IHashCodeProvider hashProvider, IComparer comparer);

[C++] public: NameValueCollection(IHashCodeProvider* hashProvider, IComparer* comparer);

[VB] Public Sub New(ByVal hashProvider As IHashCodeProvider, ByVal comparer As IComparer)

[JScript] public function NameValueCollection(hashProvider : IHashCodeProvider, comparer : IComparer);

Description

Initializes a new instance of the **System.Collections.Specialized.NameValueCollection** class that is empty, has the default initial capacity and uses the specified hash code provider and the specified comparer.

The capacity is the number of key-and-value pairs that the **System.Collections.Specialized.NameValueCollection** can contain. The default initial capacity is zero. The capacity is automatically increased as required. The **System.Collections.IHashCodeProvider** that will supply the hash codes for all keys in the **System.Collections.Specialized.NameValueCollection**. The **System.Collections.IComparer** to use to determine whether two keys are equal.

NameValueCollection

Example Syntax:

ToString


```

1
2 [C#] public NameValueCollection(int capacity, NameValueCollection col);
3 [C++] public: NameValueCollection(int capacity, NameValueCollection* col);
4 [VB] Public Sub New(ByVal capacity As Integer, ByVal col As
5 NameValueCollection)
6 [JScript] public function NameValueCollection(capacity : int, col :
7 NameValueCollection);
8

```

Description

Copies the entries from the specified **System.Collections.Specialized.NameValueCollection** to a new **System.Collections.Specialized.NameValueCollection** with the specified initial capacity or the same initial capacity as the number of entries copied, whichever is greater, and using the default case-insensitive hash code provider and the default case-insensitive comparer.

The capacity is the number of key-and-value pairs that the **System.Collections.Specialized.NameValueCollection** can contain. The default initial capacity is zero. The capacity is automatically increased as required. The initial number of entries that the **System.Collections.Specialized.NameValueCollection** can contain. The **System.Collections.Specialized.NameValueCollection** to copy to the new **System.Collections.Specialized.NameValueCollection** instance.

NameValueCollection

Example Syntax:

ToString

1
2 [C#] protected NameValueCollection(SerializationInfo info, StreamingContext
3 context);
4 [C++] protected: NameValueCollection(SerializationInfo* info, StreamingContext
5 context);
6 [VB] Protected Sub New(ByVal info As SerializationInfo, ByVal context As
7 StreamingContext)
8 [JScript] protected function NameValueCollection(info : SerializationInfo, context
9 : StreamingContext);
10

11 *Description*

12 Initializes a new instance of the
13 **System.Collections.Specialized.NameValueCollection** class that is serializable
14 and uses the specified **System.Runtime.Serialization.SerializationInfo** and
15 **System.Runtime.Serialization.StreamingContext** . A
16 **System.Runtime.Serialization.SerializationInfo** object that contains the
17 information required to serialize the new
18 **System.Collections.Specialized.NameValueCollection** instance. A
19 **System.Runtime.Serialization.StreamingContext** object that contains the source
20 and destination of the serialized stream associated with the new
21 **System.Collections.Specialized.NameValueCollection** instance.

22 NameValueCollection

23 *Example Syntax:*

24 ToString
25

```

1
2 [C#] public NameValueCollection(int capacity, IHashCodeProvider hashProvider,
3 IComparer comparer);
4 [C++] public: NameValueCollection(int capacity, IHashCodeProvider*
5 hashProvider, IComparer* comparer);
6 [VB] Public Sub New(ByVal capacity As Integer, ByVal hashProvider As
7 IHashCodeProvider, ByVal comparer As IComparer)
8 [JScript] public function NameValueCollection(capacity : int, hashProvider :
9 IHashCodeProvider, comparer : IComparer);
10

```

Description

Initializes a new instance of the **System.Collections.Specialized.NameValueCollection** class that is empty, has the specified initial capacity and uses the specified hash code provider and the specified comparer.

The capacity is the number of key-and-value pairs that the **System.Collections.Specialized.NameValueCollection** can contain. The default initial capacity is zero. The capacity is automatically increased as required. The initial number of entries that the **System.Collections.Specialized.NameValueCollection** can contain. The **System.Collections.IHashCodeProvider** that will supply the hash codes for all keys in the **System.Collections.Specialized.NameValueCollection**. The **System.Collections.IComparer** to use to determine whether two keys are equal.

AllKeys

ToString

```

1 [C#] public virtual string[] AllKeys {get;}
2
3 [C++] public: __property virtual String* get_AllKeys();
4
5 [VB] Overridable Public ReadOnly Property AllKeys As String ()
6
7 [JScript] public function get AllKeys() : String[];
8

```

Description

Gets all the keys in the

System.Collections.Specialized.NameValueCollection .

If the collection is empty, this method returns an empty **System.String** array, not **null** .

Count

IsReadOnly

Item

ToString

Description

Gets the entry at the specified index of the

System.Collections.Specialized.NameValueCollection .

This property provides the ability to access a specific element in the collection by using the following syntax: **myCollection[index]** . The zero-based index of the entry to locate in the collection.

Item

ToString

```

1
2 [C#] public string this[string name] {get; set;}
3 [C++] public: __property String* get_Item(String* name);public: __property void
4 set_Item(String* name, String*);
5 [VB] Public Default Property Item(ByVal name As String) As String
6 [JScript] returnValue =
7 NameValueCollectionObject.Item(name);NameValueCollectionObject.Item(name
8 ) = returnValue; Gets or sets the specified entry of the
9 System.Collections.Specialized.NameValueCollection .

```

Description

Gets or sets the entry with the specified key in the
System.Collections.Specialized.NameValueCollection .

This property provides the ability to access a specific element in the collection by using the following syntax: myCollection[name] . The **System.String** key of the entry to locate. The key can be **null**.

Keys

Add

```

20 [C#] public void Add(NameValueCollection c);
21 [C++] public: void Add(NameValueCollection* c);
22 [VB] Public Sub Add(ByVal c As NameValueCollection)
23 [JScript] public function Add(c : NameValueCollection); Adds entries to the
24 current System.Collections.Specialized.NameValueCollection .
25

```

1
2 *Description*

3 Copies the entries in the specified

4 **System.Collections.Specialized.NameValueCollection** to the current

5 **System.Collections.Specialized.NameValueCollection** .

6 If a key in *c* already exists in the target

7 **System.Collections.Specialized.NameValueCollection** instance, the associated

8 value in *c* is added to the existing comma-separated list of values associated with

9 the same key in the target **System.Collections.Specialized.NameValueCollection**

10 instance. The **System.Collections.Specialized.NameValueCollection** to copy to

11 the current **System.Collections.Specialized.NameValueCollection**.

12 Add

13
14 [C#] public virtual void Add(string name, string value);

15 [C++] public: virtual void Add(String* name, String* value);

16 [VB] Overridable Public Sub Add(ByVal name As String, ByVal value As String)

17 [JScript] public function Add(name : String, value : String);

18
19 *Description*

20 Adds an entry with the specified name and value to the

21 **System.Collections.Specialized.NameValueCollection** .

22 If the specified key already exists in the target

23 **System.Collections.Specialized.NameValueCollection** instance, the specified

24 value is added to the existing comma-separated list of values associated with the

25 same key in the target **System.Collections.Specialized.NameValueCollection**

1 instance. The **System.String** key of the entry to add. The key can be **null**. The
2 **System.String** value of the entry to add. The value can be **null**.

3 Clear

4
5 [C#] public void Clear();

6 [C++] public: void Clear();

7 [VB] Public Sub Clear()

8 [JScript] public function Clear();

9
10 *Description*

11 Invalidates the cached arrays and removes all entries from the
12 **System.Collections.Specialized.NameValueCollection** .

13 CopyTo

14
15 [C#] public void CopyTo(Array dest, int index);

16 [C++] public: void CopyTo(Array* dest, int index);

17 [VB] Public Sub CopyTo(ByVal dest As Array, ByVal index As Integer)

18 [JScript] public function CopyTo(dest : Array, index : int);

19
20 *Description*

21 Copies the entire **System.Collections.Specialized.NameValueCollection**
22 to a compatible one-dimensional **System.Array** , starting at the specified index of
23 the target array.

24 The specified array must be of a compatible type. The one-dimensional
25 **System.Array** that is the destination of the elements copied from

System.Collections.Specialized.NameValueCollection. The **System.Array** must have zero-based indexing. The zero-based index in *dest* at which copying begins.

Get

[C#] public virtual string Get(int index);

[C++] public: virtual String* Get(int index);

[VB] Overridable Public Function Get(ByVal index As Integer) As String

[JScript] public function Get(index : int) : String;

Description

Gets the values at the specified index of the **System.Collections.Specialized.NameValueCollection** combined into one comma-separated list.

Return Value: A **System.String** that contains a comma-separated list of the values at the specified index of the

System.Collections.Specialized.NameValueCollection , if found; otherwise, **null** . The zero-based index of the entry that contains the values to get from the collection.

Get

[C#] public virtual string Get(string name);

[C++] public: virtual String* Get(String* name);

[VB] Overridable Public Function Get(ByVal name As String) As String

[JScript] public function Get(name : String) : String; Gets the values of a specified entry in the **System.Collections.Specialized.NameValueCollection** combined

into one comma-separated list.

Description

Gets the values associated with the specified key from the **System.Collections.Specialized.NameValueCollection** combined into one comma-separated list.

Return Value: A **System.String** that contains a comma-separated list of the values associated with the specified key from the **System.Collections.Specialized.NameValueCollection** , if found; otherwise, **null** . The **System.String** key of the entry that contains the values to get. The key can be **null**.

GetKey

[C#] public virtual string GetKey(int index);

[C++] public: virtual String* GetKey(int index);

[VB] Overridable Public Function GetKey(ByVal index As Integer) As String

[JScript] public function GetKey(index : int) : String;

Description

Gets the key at the specified index of the **System.Collections.Specialized.NameValueCollection** .

Return Value: A **System.String** that contains the key at the specified index of the **System.Collections.Specialized.NameValueCollection** , if found; otherwise, **null** . The zero-based index of the key to get from the collection.

GetValues

1
2 [C#] public virtual string[] GetValues(int index);

3 [C++] public: virtual String* GetValues(int index) __gc[];

4 [VB] Overridable Public Function GetValues(ByVal index As Integer) As String()

5 [JScript] public function GetValues(index : int) : String[];

6
7 *Description*

8 Gets the values at the specified index of the

9 **System.Collections.Specialized.NameValueCollection** .

10 *Return Value:* A **System.String** array that contains the values at the specified
11 index of the **System.Collections.Specialized.NameValueCollection** , if found;
12 otherwise, **null** . The zero-based index of the entry that contains the values to get
13 from the collection.

14 **GetValues**

15
16 [C#] public virtual string[] GetValues(string name);

17 [C++] public: virtual String* GetValues(String* name) __gc[];

18 [VB] Overridable Public Function GetValues(ByVal name As String) As String()

19 [JScript] public function GetValues(name : String) : String[]; Gets the values of a
20 specified entry in the **System.Collections.Specialized.NameValueCollection** .

21
22 *Description*

23 Gets the values associated with the specified key from the

24 **System.Collections.Specialized.NameValueCollection** .

25 *Return Value:* A **System.String** array that contains the values associated with the

specified key from the **System.Collections.Specialized.NameValueCollection** , if found; otherwise, **null** . The **System.String** key of the entry that contains the values to get. The key can be **null**.

HasKeys

[C#] public bool HasKeys();

[C++] public: bool HasKeys();

[VB] Public Function HasKeys() As Boolean

[JScript] public function HasKeys() : Boolean;

Description

Gets a value indicating whether the **System.Collections.Specialized.NameValueCollection** contains keys that are not **null** .

Return Value: **true** if the **System.Collections.Specialized.NameValueCollection** contains keys that are not **null** ; otherwise, **false** .

InvalidateCachedArrays

[C#] protected void InvalidateCachedArrays();

[C++] protected: void InvalidateCachedArrays();

[VB] Protected Sub InvalidateCachedArrays()

[JScript] protected function InvalidateCachedArrays();

Description

Resets the cached arrays of the collection to **null** .

The arrays returned by **System.Collections.Specialized.NameValueCollection.AllKeys** are cached for better performance and are automatically refreshed when the collection changes. A derived class can invalidate the cached version by calling **System.Collections.Specialized.NameValueCollection.InvalidateCachedArrays**, thereby forcing the arrays to be recreated.

Remove

[C#] public virtual void Remove(string name);
[C++] public: virtual void Remove(String* name);
[VB] Overridable Public Sub Remove(ByVal name As String)
[JScript] public function Remove(name : String);

Description

Removes the entries with the specified key from the **System.Collections.Specialized.NameObjectCollectionBase** instance.

In collections such as lists, queues and stacks, the elements that follow the removed element move up to occupy the vacated spot. The **System.String** key of the entry to remove. The key can be **null**.

Set

[C#] public virtual void Set(string name, string value);
[C++] public: virtual void Set(String* name, String* value);
[VB] Overridable Public Sub Set(ByVal name As String, ByVal value As String)
[JScript] public function Set(name : String, value : String);

1
2 *Description*

3 Sets the value of an entry in the

4 **System.Collections.Specialized.NameValueCollection** .

5 If the specified key already exists in the collection, this method overwrites
6 the existing values with the specified value. To add the new value to the existing
7 list of values, use the

8 **System.Collections.Specialized.NameValueCollection.Add(System.Collections**
9 **.Specialized.NameValueCollection)** method. The **System.String** key of the entry
10 to add the new value to. The key can be **null**. The **System.Object** that represents
11 the new value to add to the specified entry. The value can be **null**.

12 BitVector32.Section structure (System.Collections.Specialized)

13 ToString
14
15

16 *Description*

17 Represents an section of the vector that can contain a integer number.

18 Use

19 **System.Collections.Specialized.BitVector32.CreateSection(System.Int16)** to
20 define a new section. A **System.Collections.Specialized.BitVector32.Section** is a
21 window into the **System.Collections.Specialized.BitVector32** and is composed of
22 the smallest number of consecutive bits that can contain the maximum value
23 specified in

24 **System.Collections.Specialized.BitVector32.CreateSection(System.Int16)** . For
25 example, a section with a maximum value of 1 is composed of only one bit,

whereas a section with a maximum value of 5 is composed of three bits. You can create a **System.Collections.Specialized.BitVector32.Section** with a maximum value of 1 to serve as a Boolean, thereby allowing you to store integers and Booleans in the same **System.Collections.Specialized.BitVector32** .

Mask

ToString

[C#] public short Mask {get;}

[C++] public: __property short get_Mask();

[VB] Public ReadOnly Property Mask As Short

[JScript] public function get Mask() : Int16;

Description

Offset

ToString

[C#] public short Offset {get;}

[C++] public: __property short get_Offset();

[VB] Public ReadOnly Property Offset As Short

[JScript] public function get Offset() : Int16;

Description

Equals

1
2 [C#] public override bool Equals(object o);

3 [C++] public: bool Equals(Object* o);

4 [VB] Overrides Public Function Equals(ByVal o As Object) As Boolean

5 [JScript] public override function Equals(o : Object) : Boolean;

6
7 *Description*

8
9 GetHashCode

10
11 [C#] public override int GetHashCode();

12 [C++] public: int GetHashCode();

13 [VB] Overrides Public Function GetHashCode() As Integer

14 [JScript] public override function GetHashCode() : int;

15
16 *Description*

17
18 ToString

19
20 [C#] public override string ToString();

21 [C++] public: String* ToString();

22 [VB] Overrides Public Function ToString() As String

23 [JScript] public override function ToString() : String;

24
25 *Description*

ToString

[C#] public static string ToString(BitVector32.Section value);

[C++] public: static String* ToString(BitVector32.Section value);

[VB] Public Shared Function ToString(ByVal value As BitVector32.Section) As

String

[JScript] public static function ToString(value : BitVector32.Section) : String;

Description

StringCollection class (System.Collections.Specialized)

ToString

Description

Represents a collection of strings.

Duplicate strings are allowed in

System.Collections.Specialized.StringCollection .

StringCollection

Example Syntax:

ToString

[C#] public StringCollection();

[C++] public: StringCollection();

1 [VB] Public Sub New()

2 [JScript] public function StringCollection();

3 Count

4 ToString

6 [C#] public int Count {get;}

7 [C++] public: __property int get_Count();

8 [VB] Public ReadOnly Property Count As Integer

9 [JScript] public function get Count() : int;

11 *Description*

12 Gets the number of strings contained in the

13 **System.Collections.Specialized.StringCollection** .

14 IsReadOnly

15 ToString

17 [C#] public bool IsReadOnly {get;}

18 [C++] public: __property bool get_IsReadOnly();

19 [VB] Public ReadOnly Property IsReadOnly As Boolean

20 [JScript] public function get IsReadOnly() : Boolean;

22 *Description*

23 Gets a value indicating whether the

24 **System.Collections.Specialized.StringCollection** is read-only.

System.Collections.Specialized.StringCollection implements the **System.Collections.Specialized.StringCollection.IsReadOnly** property because it is required by the interface.

IsSynchronized

ToString

[C#] public bool IsSynchronized {get;}

[C++] public: __property bool get_IsSynchronized();

[VB] Public ReadOnly Property IsSynchronized As Boolean

[JScript] public function get IsSynchronized() : Boolean;

Description

Gets a value indicating whether access to the **System.Collections.Specialized.StringCollection** is synchronized (thread-safe).

System.Collections.Specialized.StringCollection implements the **System.Collections.Specialized.StringCollection.IsSynchronized** property because it is required by the interface.

Item

ToString

[C#] public string this[int index] {get; set;}

[C++] public: __property String* get_Item(int index);public: __property void set_Item(int index, String*);

[VB] Public Default Property Item(ByVal index As Integer) As String

[JScript] returnValue =

StringCollectionObject.Item(index);StringCollectionObject.Item(index) =
returnValue;

Description

Gets or sets the element at the specified index.

This property provides the ability to access a specific element in the collection by using the following syntax: myCollection[index] . The zero-based index of the entry to get or set.

SyncRoot

ToString

[C#] public object SyncRoot {get;}

[C++] public: __property Object* get_SyncRoot();

[VB] Public ReadOnly Property SyncRoot As Object

[JScript] public function get SyncRoot() : Object;

Description

Gets an object that can be used to synchronize access to the **System.Collections.Specialized.StringCollection** .

Derived classes can provide their own synchronized version of the **System.Collections.Specialized.StringCollection** using the **System.Collections.Specialized.StringCollection.SyncRoot** property. The synchronizing code must perform operations on the **System.Collections.Specialized.StringCollection.SyncRoot** of the **System.Collections.Specialized.StringCollection** , not directly on the

System.Collections.Specialized.StringCollection . This ensures proper operation of collections that are derived from other objects. Specifically, it maintains proper synchronization with other threads that might be simultaneously modifying the **System.Collections.Specialized.StringCollection** object.

Add

[C#] public int Add(string value);

[C++] public: int Add(String* value);

[VB] Public Function Add(ByVal value As String) As Integer

[JScript] public function Add(value : String) : int;

Description

Adds a string to the end of the

System.Collections.Specialized.StringCollection .

Return Value: The zero-based index at which the new element is inserted.

Duplicate strings are allowed in

System.Collections.Specialized.StringCollection . The string to add to the end of the **System.Collections.Specialized.StringCollection** .

AddRange

[C#] public void AddRange(string[] value);

[C++] public: void AddRange(String* value __gc[]);

[VB] Public Sub AddRange(ByVal value() As String)

[JScript] public function AddRange(value : String[]);

1
2 *Description*

3 Copies the elements of a string array to the end of the

4 **System.Collections.Specialized.StringCollection** .

5 Duplicate strings are allowed in

6 **System.Collections.Specialized.StringCollection** . An array of strings to add to

7 the end of the **System.Collections.Specialized.StringCollection** .

8 Clear

9
10 [C#] public void Clear();

11 [C++] public: __sealed void Clear();

12 [VB] NotOverridable Public Sub Clear()

13 [JScript] public function Clear();

14
15 *Description*

16 Removes all the strings from the

17 **System.Collections.Specialized.StringCollection** .

18 **System.Collections.Specialized.StringCollection.Count** is set to zero.

19 Contains

20
21 [C#] public bool Contains(string value);

22 [C++] public: bool Contains(String* value);

23 [VB] Public Function Contains(ByVal value As String) As Boolean

24 [JScript] public function Contains(value : String) : Boolean;

Description

Determines whether the specified string is in the

System.Collections.Specialized.StringCollection .

Return Value: **true** if *value* is found in the

System.Collections.Specialized.StringCollection ; otherwise, **false** .

The

System.Collections.Specialized.StringCollection.Contains(System.String)

method can confirm the existence of a string before performing further operations.

The string to locate in the **System.Collections.Specialized.StringCollection** .

CopyTo

[C#] public void CopyTo(string[] array, int index);

[C++] public: void CopyTo(String* array __gc[], int index);

[VB] Public Sub CopyTo(ByVal array() As String, ByVal index As Integer)

[JScript] public function CopyTo(array : String[], index : int);

Description

Copies the **System.Collections.Specialized.StringCollection** values to a compatible one-dimensional **System.Array** , starting at the specified index of the target array.

The specified array must be of a compatible type. The one-dimensional **System.Array** that is the destination of the values copied from **System.Collections.Specialized.StringCollection**. The **System.Array** must have zero-based indexing. The zero-based index in *array* at which copying begins.

GetEnumerator

```
[C#] public StringEnumerator GetEnumerator();  
[C++] public: StringEnumerator* GetEnumerator();  
[VB] Public Function GetEnumerator() As StringEnumerator  
[JScript] public function GetEnumerator() : StringEnumerator;
```

Description

Returns an enumerator that can iterate through the
System.Collections.Specialized.StringCollection .

Return Value: An **System.Collections.IEnumerator** for the
System.Collections.Specialized.StringCollection .

Enumerators are intended to be used only to read data in the collection.
Enumerators cannot be used to modify the underlying collection.

IndexOf

```
[C#] public int IndexOf(string value);  
[C++] public: int IndexOf(String* value);  
[VB] Public Function IndexOf(ByVal value As String) As Integer  
[JScript] public function IndexOf(value : String) : int;
```

Description

Searches for the specified string and returns the zero-based index of the
first occurrence within the **System.Collections.Specialized.StringCollection** .

Return Value: The zero-based index of the first occurrence of *value* in the **System.Collections.Specialized.StringCollection** , if found; otherwise, -1.

This method performs a linear search. On average, this is an $O(n/2)$ operation, where n is **System.Collections.Specialized.StringCollection.Count** .

The longest search is an $O(n)$ operation, where n is **System.Collections.Specialized.StringCollection.Count** . The string to locate.

Insert

[C#] public void Insert(int index, string value);

[C++] public: void Insert(int index, String* value);

[VB] Public Sub Insert(ByVal index As Integer, ByVal value As String)

[JScript] public function Insert(index : int, value : String);

Description

Inserts a string into the **System.Collections.Specialized.StringCollection** at the specified index.

Duplicate strings are allowed in **System.Collections.Specialized.StringCollection** . The zero-based index at which *value* is inserted. The string to insert.

Remove

[C#] public void Remove(string value);

[C++] public: void Remove(String* value);

[VB] Public Sub Remove(ByVal value As String)

[JScript] public function Remove(value : String);

1
2 *Description*

3 Removes the first occurrence of a specific string from the

4 **System.Collections.Specialized.StringCollection** .

5 Duplicate strings are allowed in

6 **System.Collections.Specialized.StringCollection** . Only the first occurrence is

7 removed. To remove all occurrences of the specified string, use

8 `RemoveAt(IndexOf(value))` repeatedly while

9 **System.Collections.Specialized.StringCollection.IndexOf(System.String)** does

10 not return -1. The string to remove from the

11 **System.Collections.Specialized.StringCollection**.

12 `RemoveAt`

13
14 [C#] `public void RemoveAt(int index);`

15 [C++] `public: __sealed void RemoveAt(int index);`

16 [VB] `NotOverridable Public Sub RemoveAt(ByVal index As Integer)`

17 [JScript] `public function RemoveAt(index : int);`

18
19 *Description*

20 Removes the string at the specified index of the

21 **System.Collections.Specialized.StringCollection** .

22 In collections such as lists, queues and stacks, the elements that follow the
23 removed element move up to occupy the vacated spot. The zero-based index of the
24 string to remove.

25 `ICollection.CopyTo`

```

1
2 [C#] void ICollection.CopyTo(Array array, int index);
3 [C++] void ICollection::CopyTo(Array* array, int index);
4 [VB] Sub CopyTo(ByVal array As Array, ByVal index As Integer) Implements
5 ICollection.CopyTo
6 [JScript] function ICollection.CopyTo(array : Array, index : int);
7     IEnumerable.GetEnumerator
8
9 [C#] IEnumerator IEnumerable.GetEnumerator();
10 [C++] IEnumerator* IEnumerable::GetEnumerator();
11 [VB] Function GetEnumerator() As IEnumerator Implements
12 IEnumerable.GetEnumerator
13 [JScript] function IEnumerable.GetEnumerator() : IEnumerator;
14     IList.Add
15
16 [C#] int IList.Add(object value);
17 [C++] int IList::Add(Object* value);
18 [VB] Function Add(ByVal value As Object) As Integer Implements IList.Add
19 [JScript] function IList.Add(value : Object) : int;
20     IList.Contains
21
22 [C#] bool IList.Contains(object value);
23 [C++] bool IList::Contains(Object* value);
24 [VB] Function Contains(ByVal value As Object) As Boolean Implements
25

```

1 IList.Contains

2 [JScript] function IList.Contains(value : Object) : Boolean;

3 IList.IndexOf

4
5 [C#] int IList.IndexOf(object value);

6 [C++] int IList::IndexOf(Object* value);

7 [VB] Function IndexOf(ByVal value As Object) As Integer Implements

8 IList.IndexOf

9 [JScript] function IList.IndexOf(value : Object) : int;

10 IList.Insert

11
12 [C#] void IList.Insert(int index, object value);

13 [C++] void IList::Insert(int index, Object* value);

14 [VB] Sub Insert(ByVal index As Integer, ByVal value As Object) Implements

15 IList.Insert

16 [JScript] function IList.Insert(index : int, value : Object);

17 IList.Remove

18
19 [C#] void IList.Remove(object value);

20 [C++] void IList::Remove(Object* value);

21 [VB] Sub Remove(ByVal value As Object) Implements IList.Remove

22 [JScript] function IList.Remove(value : Object);

23 StringDictionary class (System.Collections.Specialized)

24 ToString

1
2
3 *Description*

4 Implements a hashtable with the key strongly typed to be a string rather
5 than an object.

6 The key is handled in a case-insensitive manner; it will be translated to
7 lower case before it is used with the string dictionary.

8 StringDictionary

9 *Example Syntax:*

10 ToString

11
12 [C#] public StringDictionary();

13 [C++] public: StringDictionary();

14 [VB] Public Sub New()

15 [JScript] public function StringDictionary();

16
17 *Description*

18 Initializes a new instance of the
19 **System.Collections.Specialized.StringDictionary** class.

20 Count

21 ToString

22
23 [C#] public virtual int Count {get;}

24 [C++] public: __property virtual int get_Count();

25 [VB] Overridable Public ReadOnly Property Count As Integer

1 [JScript] public function get Count() : int;

3 *Description*

4 Gets the number of key-and-value pairs in the

5 **System.Collections.Specialized.StringDictionary** .

6 IsSynchronized

7 ToString

9 [C#] public virtual bool IsSynchronized {get;}

10 [C++] public: __property virtual bool get_IsSynchronized();

11 [VB] Overridable Public ReadOnly Property IsSynchronized As Boolean

12 [JScript] public function get IsSynchronized() : Boolean;

14 *Description*

15 Indicates whether access to the

16 **System.Collections.Specialized.StringDictionary** is synchronized (thread-safe).

17 This property is read-only.

18 Item

19 ToString

21 [C#] public virtual string this[string key] {get; set;}

22 [C++] public: __property virtual String* get_Item(String* key);public: __property
23 virtual void set_Item(String* key, String*);

24 [VB] Overridable Public Default Property Item(ByVal key As String) As String

25 [JScript] returnValue =

StringDictionaryObject.Item(key);StringDictionaryObject.Item(key) =
return Value;

Description

Gets or sets the value associated with the specified key.

The key is handled in a case-insensitive manner; it will be translated to lower case before it is used. The key whose value to get or set.

Keys

ToString

[C#] public virtual ICollection Keys {get;}

[C++] public: __property virtual ICollection* get_Keys();

[VB] Overridable Public ReadOnly Property Keys As ICollection

[JScript] public function get Keys() : ICollection;

Description

Gets a collection of keys in the

System.Collections.Specialized.StringDictionary .

The order of the keys in the **System.Collections.ICollection** is unspecified, but it is the same order as the associated values in the

System.Collections.ICollection returned by the

System.Collections.Specialized.StringDictionary.Values method.

SyncRoot

ToString

```

1
2 [C#] public virtual object SyncRoot {get;}
3 [C++] public: __property virtual Object* get_SyncRoot();
4 [VB] Overridable Public ReadOnly Property SyncRoot As Object
5 [JScript] public function get SyncRoot() : Object;
6

```

Description

Gets an object that can be used to synchronize access to the

System.Collections.Specialized.StringDictionary .

Values

ToString

```

13 [C#] public virtual ICollection Values {get;}
14 [C++] public: __property virtual ICollection* get_Values();
15 [VB] Overridable Public ReadOnly Property Values As ICollection
16 [JScript] public function get Values() : ICollection;
17

```

Description

Gets a collection of values in the

System.Collections.Specialized.StringDictionary .

The order of the values in the **System.Collections.ICollection** is unspecified, but it is the same order as the associated keys in the **System.Collections.ICollection** returned by the **System.Collections.Specialized.StringDictionary.Keys** method.

Add

1
2 [C#] public virtual void Add(string key, string value);

3 [C++] public: virtual void Add(String* key, String* value);

4 [VB] Overridable Public Sub Add(ByVal key As String, ByVal value As String)

5 [JScript] public function Add(key : String, value : String);

6
7 *Description*

8 Adds an entry with the specified key and value into the

9 **System.Collections.Specialized.StringDictionary** .

10 The key is handled in a case-insensitive manner; it will be translated to
11 lower case before it is added to the string dictionary. The key of the entry to add.

12 The value of the entry to add.

13 **Clear**

14
15 [C#] public virtual void Clear();

16 [C++] public: virtual void Clear();

17 [VB] Overridable Public Sub Clear()

18 [JScript] public function Clear();

19
20 *Description*

21 Removes all entries from the

22 **System.Collections.Specialized.StringDictionary** .

23 **ContainsKey**

24
25 [C#] public virtual bool ContainsKey(string key);

1 [C++] public: virtual bool ContainsKey(String* key);

2 [VB] Overridable Public Function ContainsKey(ByVal key As String) As Boolean

3 [JScript] public function ContainsKey(key : String) : Boolean;

4
5 *Description*

6 Determines if the string dictionary contains a specific key

7 *Return Value:* **true** if the **System.Collections.Specialized.StringDictionary**
8 contains an entry with the specified key; otherwise, **false** .

9 This implementation is an O(1) operation. The key to locate in the
10 **System.Collections.Specialized.StringDictionary**.

11 **ContainsValue**

12
13 [C#] public virtual bool ContainsValue(string value);

14 [C++] public: virtual bool ContainsValue(String* value);

15 [VB] Overridable Public Function ContainsValue(ByVal value As String) As
16 Boolean

17 [JScript] public function ContainsValue(value : String) : Boolean;

18
19 *Description*

20 Determines if the **System.Collections.Specialized.StringDictionary**
21 contains a specific value.

22 *Return Value:* **true** if the **System.Collections.Specialized.StringDictionary**
23 contains an element with the specified value; otherwise, **false** .

The values of the elements of the **StringDictionary** are compared to the specified value using the **System.Object.Equals(System.Object)** method. The value to locate in the **System.Collections.Specialized.StringDictionary**.

CopyTo

[C#] public virtual void CopyTo(Array array, int index);

[C++] public: virtual void CopyTo(Array* array, int index);

[VB] Overridable Public Sub CopyTo(ByVal array As Array, ByVal index As Integer)

[JScript] public function CopyTo(array : Array, index : int);

Description

Copies the string dictionary values to a one-dimensional **System.Array** instance at the specified index.

System.Collections.Specialized.StringDictionary.CopyTo(System.Array, System.Int32) only copies the values in the **StringDictionary**, not the keys. The one-dimensional **System.Array** that is the destination of the values copied from the **System.Collections.Specialized.StringDictionary**. The index in the array where copying begins.

GetEnumerator

[C#] public virtual IEnumerator GetEnumerator();

[C++] public: virtual IEnumerator* GetEnumerator();

[VB] Overridable Public Function GetEnumerator() As IEnumerator

[JScript] public function GetEnumerator() : IEnumerator;

Description

Returns an enumerator that can iterate through the string dictionary.

Return Value: An **System.Collections.IEnumerator** that can iterate through the string dictionary.

The enumerator does not have exclusive access to the **System.Collections.Specialized.StringDictionary**; therefore, any changes made to the **System.Collections.Specialized.StringDictionary** can cause **System.Collections.IEnumerator.Current** or **System.Collections.IEnumerator.MoveNext** to throw an exception.

Remove

[C#] public virtual void Remove(string key);

[C++] public: virtual void Remove(String* key);

[VB] Overridable Public Sub Remove(ByVal key As String)

[JScript] public function Remove(key : String);

Description

Removes the entry with the specified key from the string dictionary.

The key is handled in a case-insensitive manner; it will be translated to lower case before it is used to find the entry to remove from the string dictionary.

The key of the entry to remove.

StringEnumerator class (System.Collections.Specialized)

ToString

1
2
3 *Description*

4 Supports a simple iteration over a

5 **System.Collections.Specialized.StringCollection .**

6 Enumerators are intended to be used only to read data in the collection.

7 Enumerators cannot be used to modify the underlying collection.

8 Current

9 ToString

10
11 [C#] public string Current {get;}

12 [C++] public: __property String* get_Current();

13 [VB] Public ReadOnly Property Current As String

14 [JScript] public function get Current() : String;

15
16 *Description*

17 Gets the current element in the collection.

18 After an enumerator is created or after a

19 **System.Collections.Specialized.StringEnumerator.Reset ,**

20 **System.Collections.Specialized.StringEnumerator.MoveNext** must be called to

21 advance the enumerator to the first element of the collection before reading the

22 value of **System.Collections.Specialized.StringEnumerator.Current** ;

23 otherwise, **System.Collections.Specialized.StringEnumerator.Current** is

24 undefined.

25 MoveNext

1
2 [C#] public bool MoveNext();

3 [C++] public: bool MoveNext();

4 [VB] Public Function MoveNext() As Boolean

5 [JScript] public function MoveNext() : Boolean;

6
7 *Description*

8 Advances the enumerator to the next element of the collection.

9 *Return Value:* **true** if the enumerator was successfully advanced to the next
10 element; **false** if the enumerator has passed the end of the collection.

11 After an enumerator is created or after a call to
12 **System.Collections.Specialized.StringEnumerator.Reset** , an enumerator is
13 positioned before the first element of the collection, and the first call to
14 **System.Collections.Specialized.StringEnumerator.MoveNext** moves the
15 enumerator over the first element of the collection.

16 **Reset**

17
18 [C#] public void Reset();

19 [C++] public: void Reset();

20 [VB] Public Sub Reset()

21 [JScript] public function Reset();

22
23
24 **SYSTEM.GLOBALIZATION NAMESPACE**
25

1 The System.Globalization namespace contains classes that define culture-
2 related information, such as the language, the country/region, the calendars in use,
3 the format patterns for dates, currency and numbers, and the sort order for strings.

4 The .NET Framework is introducing a distinction to the globalization
5 world. The concept that was previously referred to as "Locale" has been split
6 apart into two separate types that allows much more flexibility. The Locale
7 concept is represented as two different types in the .NET world: CultureInfo and
8 RegionInfo. CultureInfo represents information about the users' culture, what
9 language they specify, how they prefer numbers formatted, what calendar they
10 use, etc. The RegionInfo class represents information about where a person
11 physically is, what currency symbol they use, if they use metric or not, etc.

12 The following is a more detailed description of the System.Globalization
13 namespace, identifying various classes, interfaces, enumerations, and so forth
14 contained in the System.Globalization namespace.

15 **System.Globalization**

16 The namespace contains classes that define culture-related information,
17 including the language, the country/region, the calendars in use, the format
18 patterns for dates, currency and numbers, and the sort order for strings.

19 *Description*

20
21 The **System.Globalization** namespace contains classes that define culture-
22 related information, including the language, the country/region, the calendars in
23 use, the format patterns for dates, currency and numbers, and the sort order for
24 strings.

25 Calendar class (System.Globalization)

1
2
3 *Description*

4 Represents time in divisions, such as weeks, months, and years.

5 A calendar divides time into measures, such as weeks, months, and years.

6 The number, length, and start of the divisions vary in each calendar.

7
8 [C#] public const int CurrentEra;

9 [C++] public: const int CurrentEra;

10 [VB] Public Const CurrentEra As Integer

11 [JScript] public var CurrentEra : int;

12
13 *Description*

14 Represents the current era for the current calendar.

15 Constructors:

16 Calendar

17 *Example Syntax:*

18
19 [C#] protected Calendar();

20 [C++] protected: Calendar();

21 [VB] Protected Sub New()

22 [JScript] protected function Calendar();

23
24 *Description*

25 Initializes a new instance of the **System.Globalization.Calendar** class.

Properties:

Eras

[C#] public abstract int[] Eras {get;}

[C++] public: __property virtual int get_Eras() = 0;

[VB] MustOverride Public ReadOnly Property Eras As Integer ()

[JScript] public abstract function get Eras() : int[];

Description

When implemented by a derived class, gets the list of eras in the current calendar.

TwoDigitYearMax

[C#] public virtual int TwoDigitYearMax {get; set;}

[C++] public: __property virtual int get_TwoDigitYearMax();public: __property virtual void set_TwoDigitYearMax(int);

[VB] Overridable Public Property TwoDigitYearMax As Integer

[JScript] public function get TwoDigitYearMax() : int;public function set TwoDigitYearMax(int);

Description

Gets or sets the last year of a 100-year range that can be represented by a 2-digit year.

This property allows a 2-digit year to be properly translated to a 4-digit year. For example, if this property is set to 2029, the 100-year range is from 1930

to 2029; therefore, a 2-digit value of 30 is interpreted as 1930, while a 2-digit value of 29 is interpreted as 2029.

Methods:

AddDays

[C#] public virtual DateTime AddDays(DateTime time, int days);

[C++] public: virtual DateTime AddDays(DateTime time, int days);

[VB] Overridable Public Function AddDays(ByVal time As DateTime, ByVal days As Integer) As DateTime

[JScript] public function AddDays(time : DateTime, days : int) : DateTime;

Description

Returns a **System.DateTime** that is the specified number of days away from the specified **System.DateTime**.

Return Value: The **System.DateTime** that results from adding the specified number of days to the specified **System.DateTime**.

The *days* value is rounded to the nearest millisecond before it is added to the specified **System.DateTime**. If *days* is negative, the resulting **System.DateTime** would be earlier than the specified **System.DateTime**. The **System.DateTime** instance to add. The number of days to add.

AddHours

[C#] public virtual DateTime AddHours(DateTime time, int hours);

[C++] public: virtual DateTime AddHours(DateTime time, int hours);

[VB] Overridable Public Function AddHours(ByVal time As DateTime, ByVal

hours As Integer) As DateTime

[JScript] public function AddHours(time : DateTime, hours : int) : DateTime;

Description

Returns a **System.DateTime** that is the specified number of hours away from the specified **System.DateTime**.

Return Value: The **System.DateTime** that results from adding the specified number of hours to the specified **System.DateTime**.

The *hours* value is rounded to the nearest millisecond before it is added to the specified **System.DateTime**. If *hours* is negative, the resulting **System.DateTime** would be earlier than the specified **System.DateTime**. The **System.DateTime** instance to add. The number of hours to add.

AddMilliseconds

[C#] public virtual DateTime AddMilliseconds(DateTime time, double milliseconds);

[C++] public: virtual DateTime AddMilliseconds(DateTime time, double milliseconds);

[VB] Overridable Public Function AddMilliseconds(ByVal time As DateTime, ByVal milliseconds As Double) As DateTime

[JScript] public function AddMilliseconds(time : DateTime, milliseconds : double) : DateTime;

Description

1 Returns a **System.DateTime** that is the specified number of milliseconds
2 away from the specified **System.DateTime** .

3 *Return Value:* The **System.DateTime** that results from adding the specified
4 number of milliseconds to the specified **System.DateTime** .

5 The *milliseconds* value is rounded to the nearest integer before it is added
6 to the specified **System.DateTime** . If *milliseconds* is negative, the resulting
7 **System.DateTime** would be earlier than the specified **System.DateTime** . The
8 **System.DateTime** instance to add. The number of milliseconds to add.

9 AddMinutes

10
11 [C#] public virtual DateTime AddMinutes(DateTime time, int minutes);

12 [C++] public: virtual DateTime AddMinutes(DateTime time, int minutes);

13 [VB] Overridable Public Function AddMinutes(ByVal time As DateTime, ByVal
14 minutes As Integer) As DateTime

15 [JScript] public function AddMinutes(time : DateTime, minutes : int) : DateTime;

16 17 Description

18 Returns a **System.DateTime** that is the specified number of minutes away
19 from the specified **System.DateTime** .

20 *Return Value:* The **System.DateTime** that results from adding the specified
21 number of minutes to the specified **System.DateTime** .

22 The *minutes* value is rounded to the nearest millisecond before it is added
23 to the specified **System.DateTime** . If *minutes* is negative, the resulting
24 **System.DateTime** would be earlier than the specified **System.DateTime** . The
25 **System.DateTime** instance to add. The number of minutes to add.

AddMonths

```
[C#] public abstract DateTime AddMonths(DateTime time, int months);  
[C++] public: virtual DateTime AddMonths(DateTime time, int months) = 0;  
[VB] MustOverride Public Function AddMonths(ByVal time As DateTime,  
ByVal months As Integer) As DateTime  
[JScript] public abstract function AddMonths(time : DateTime, months : int) :  
DateTime;
```

Description

When implemented by a derived class, returns a **System.DateTime** that is the specified number of months away from the specified **System.DateTime**.

Return Value: The **System.DateTime** that results from adding the specified number of months to the specified **System.DateTime**.

The year part of the resulting **System.DateTime** is affected if the resulting month is beyond the last month of the current year. The day part of the resulting **System.DateTime** is also affected if the resulting day is not a valid day in the resulting month of the resulting year; it is changed to the last valid day in the resulting month of the resulting year. The time-of-day part of the resulting **System.DateTime** remains the same as the specified **System.DateTime**. The **System.DateTime** instance to add. The number of months to add.

AddSeconds

```
[C#] public virtual DateTime AddSeconds(DateTime time, int seconds);  
[C++] public: virtual DateTime AddSeconds(DateTime time, int seconds);
```

1 [VB] Overridable Public Function AddSeconds(ByVal time As DateTime, ByVal
2 seconds As Integer) As DateTime

3 [JScript] public function AddSeconds(time : DateTime, seconds : int) : DateTime;

4
5 *Description*

6 Returns a **System.DateTime** that is the specified number of seconds away
7 from the specified **System.DateTime** .

8 *Return Value:* The **System.DateTime** that results from adding the specified
9 number of seconds to the specified **System.DateTime** .

10 The *value* parameter is rounded to the nearest millisecond before it is added
11 to the specified **System.DateTime** . If *value* is negative, the resulting
12 **System.DateTime** would be earlier than the specified **System.DateTime** . The
13 **System.DateTime** instance to add. The number of seconds to add.

14 **AddWeeks**

15
16 [C#] public virtual DateTime AddWeeks(DateTime time, int weeks);

17 [C++] public: virtual DateTime AddWeeks(DateTime time, int weeks);

18 [VB] Overridable Public Function AddWeeks(ByVal time As DateTime, ByVal
19 weeks As Integer) As DateTime

20 [JScript] public function AddWeeks(time : DateTime, weeks : int) : DateTime;

21
22 *Description*

23 Returns a **System.DateTime** that is the specified number of weeks away
24 from the specified **System.DateTime** .
25

Return Value: The **System.DateTime** that results from adding the specified number of weeks to the specified **System.DateTime** .

If *weeks* is negative, the resulting **System.DateTime** would be earlier than the specified **System.DateTime** . The **System.DateTime** instance to add. The number of weeks to add.

AddYears

[C#] public abstract DateTime AddYears(DateTime time, int years);

[C++] public: virtual DateTime AddYears(DateTime time, int years) = 0;

[VB] MustOverride Public Function AddYears(ByVal time As DateTime, ByVal years As Integer) As DateTime

[JScript] public abstract function AddYears(time : DateTime, years : int) : DateTime;

Description

When implemented by a derived class, returns a **System.DateTime** that is the specified number of years away from the specified **System.DateTime** .

Return Value: The **System.DateTime** that results from adding the specified number of years to the specified **System.DateTime** .

The day part of the resulting **System.DateTime** is affected if the resulting day is not a valid day in the resulting month of the resulting year; it is changed to the last valid day in the resulting month of the resulting year. The time-of-day part of the resulting **System.DateTime** remains the same as the specified **System.DateTime** . The **System.DateTime** instance to add. The number of years to add.

GetDayOfMonth

[C#] public abstract int GetDayOfMonth(DateTime time);

[C++] public: virtual int GetDayOfMonth(DateTime time) = 0;

[VB] MustOverride Public Function GetDayOfMonth(ByVal time As DateTime)

As Integer

[JScript] public abstract function GetDayOfMonth(time : DateTime) : int;

Description

When implemented by a derived class, gets the day of the month in the specified **System.DateTime** .

Return Value: An integer that represents the day of the month in *time* . The **System.DateTime** instance to read.

GetDayOfWeek

[C#] public abstract DayOfWeek GetDayOfWeek(DateTime time);

[C++] public: virtual DayOfWeek GetDayOfWeek(DateTime time) = 0;

[VB] MustOverride Public Function GetDayOfWeek(ByVal time As DateTime)

As DayOfWeek

[JScript] public abstract function GetDayOfWeek(time : DateTime) : DayOfWeek;

Description

When implemented by a derived class, gets the day of the week in the specified **System.DateTime** .

1 *Return Value:* A **System.DayOfWeek** value that represents the day of the week in
2 *time* .

3 The **System.DayOfWeek** values are Sunday, Monday, Tuesday,
4 Wednesday, Thursday, Friday, and Saturday. The **System.DateTime** instance to
5 read.

6 **GetDayOfYear**

7
8 [C#] public abstract int GetDayOfYear(DateTime time);

9 [C++] public: virtual int GetDayOfYear(DateTime time) = 0;

10 [VB] MustOverride Public Function GetDayOfYear(ByVal time As DateTime) As
11 Integer

12 [JScript] public abstract function GetDayOfYear(time : DateTime) : int;

13
14 *Description*

15 When implemented by a derived class, gets the day of the year in the
16 specified **System.DateTime** .

17 *Return Value:* An integer that represents the day of the year in *time* . The
18 **System.DateTime** instance to read.

19 **GetDaysInMonth**

20
21 [C#] public virtual int GetDaysInMonth(int year, int month);

22 [C++] public: virtual int GetDaysInMonth(int year, int month);

23 [VB] Overridable Public Function GetDaysInMonth(ByVal year As Integer,
24 ByVal month As Integer) As Integer

25 [JScript] public function GetDaysInMonth(year : int, month : int) : int; Gets the

number of days in the specified month.

Description

Gets the number of days in the month specified by the *year* and *month* parameters.

Return Value: The number of days in the specified month in the specified year in the current era.

For example, in the Gregorian calendar, this method might return 28 or 29 for February (*month* = 2), depending on whether *year* is a leap year. An integer that represents the year. An integer that represents the month.

GetDaysInMonth

[C#] public abstract int GetDaysInMonth(int year, int month, int era);

[C++] public: virtual int GetDaysInMonth(int year, int month, int era) = 0;

[VB] MustOverride Public Function GetDaysInMonth(ByVal year As Integer, ByVal month As Integer, ByVal era As Integer) As Integer

[JScript] public abstract function GetDaysInMonth(year : int, month : int, era : int) : int;

Description

When implemented by a derived class, gets the number of days in the month specified by the *year* , *month* , and *era* parameters.

Return Value: The number of days in the specified month in the specified year in the specified era.

For example, in the Gregorian calendar, this method might return 28 or 29 for February (*month* = 2), depending on whether *year* is a leap year. An integer that represents the year. An integer that represents the month. An integer that represents the era.

GetDaysInYear

[C#] public virtual int GetDaysInYear(int year);

[C++] public: virtual int GetDaysInYear(int year);

[VB] Overridable Public Function GetDaysInYear(ByVal year As Integer) As Integer

[JScript] public function GetDaysInYear(year : int) : int; Gets the number of days in the specified year.

Description

Gets the number of days in the year specified by the *year* parameter.

Return Value: The number of days in the specified year in the current era.

For example, in the Gregorian calendar, this method might return 365 or 366, depending on whether *year* is a leap year. An integer that represents the year.

GetDaysInYear

[C#] public abstract int GetDaysInYear(int year, int era);

[C++] public: virtual int GetDaysInYear(int year, int era) = 0;

[VB] MustOverride Public Function GetDaysInYear(ByVal year As Integer, ByVal era As Integer) As Integer

[JScript] public abstract function GetDaysInYear(year : int, era : int) : int;

Description

When implemented by a derived class, gets the number of days in the year specified by the *year* and *era* parameters.

Return Value: The number of days in the specified year in the specified era.

For example, in the Gregorian calendar, this method might return 365 or 366, depending on whether *year* is a leap year. An integer that represents the year.
An integer that represents the era.

GetEra

[C#] public abstract int GetEra(DateTime time);

[C++] public: virtual int GetEra(DateTime time) = 0;

[VB] MustOverride Public Function GetEra(ByVal time As DateTime) As Integer

[JScript] public abstract function GetEra(time : DateTime) : int;

Description

When implemented by a derived class, gets the era in the specified **System.DateTime** instance.

Return Value: An integer that represents the era in *time* . The **System.DateTime** instance to read.

GetHour

[C#] public virtual int GetHour(DateTime time);

[C++] public: virtual int GetHour(DateTime time);

[VB] Overridable Public Function GetHour(ByVal time As DateTime) As Integer

1 [JScript] public function GetHour(time : DateTime) : int;

2
3 *Description*

4 Gets the hours value in the specified **System.DateTime** .

5 *Return Value:* An integer from 0 to 23 that represents the hour in *time* . The
6 **System.DateTime** instance to read.

7 GetMilliseconds

8
9 [C#] public virtual double GetMilliseconds(DateTime time);

10 [C++] public: virtual double GetMilliseconds(DateTime time);

11 [VB] Overridable Public Function GetMilliseconds(ByVal time As DateTime) As
12 Double

13 [JScript] public function GetMilliseconds(time : DateTime) : double;

14
15 *Description*

16 Gets the milliseconds value in the specified **System.DateTime** .

17 *Return Value:* An integer that represents the milliseconds in *time* .

18 The returned value is an integer from 0 to 999. The **System.DateTime**
19 instance to read.

20 GetMinute

21
22 [C#] public virtual int GetMinute(DateTime time);

23 [C++] public: virtual int GetMinute(DateTime time);

24 [VB] Overridable Public Function GetMinute(ByVal time As DateTime) As
25 Integer

1 [JScript] public function GetMinute(time : DateTime) : int;

3 *Description*

4 Gets the minutes value in the specified **System.DateTime** .

5 *Return Value:* An integer that represents the minutes in *time* .

6 The returned value is an integer from 0 to 59. The **System.DateTime**
7 instance to read.

8 GetMonth

10 [C#] public abstract int GetMonth(DateTime time);

11 [C++] public: virtual int GetMonth(DateTime time) = 0;

12 [VB] MustOverride Public Function GetMonth(ByVal time As DateTime) As

13 Integer

14 [JScript] public abstract function GetMonth(time : DateTime) : int;

16 *Description*

17 When implemented by a derived class, gets the month in the specified
18 **System.DateTime** .

19 *Return Value:* An integer that represents the month in *time* . The

20 **System.DateTime** instance to read.

21 GetMonthsInYear

23 [C#] public virtual int GetMonthsInYear(int year);

24 [C++] public: virtual int GetMonthsInYear(int year);

25 [VB] Overridable Public Function GetMonthsInYear(ByVal year As Integer) As

Integer

[JScript] public function GetMonthsInYear(year : int) : int; Gets the number of months in the specified year.

Description

Gets the number of months in the year specified by the *year* parameter.

Return Value: The number of months in the specified year in the current era. An integer that represents the year.

GetMonthsInYear

[C#] public abstract int GetMonthsInYear(int year, int era);

[C++] public: virtual int GetMonthsInYear(int year, int era) = 0;

[VB] MustOverride Public Function GetMonthsInYear(ByVal year As Integer, ByVal era As Integer) As Integer

[JScript] public abstract function GetMonthsInYear(year : int, era : int) : int;

Description

When implemented by a derived class, gets the number of months in the year specified by the *year* and *era* parameters.

Return Value: The number of months in the specified year in the specified era. An integer that represents the year. An integer that represents the era.

GetSecond

[C#] public virtual int GetSecond(DateTime time);

[C++] public: virtual int GetSecond(DateTime time);

1 [VB] Overridable Public Function GetSecond(ByVal time As DateTime) As
2 Integer

3 [JScript] public function GetSecond(time : DateTime) : int;

4
5 *Description*

6 Gets the seconds value in the specified **System.DateTime** .

7 *Return Value:* An integer that represents the seconds in *time* .

8 The returned value is an integer from 0 to 59. The **System.DateTime**
9 instance to read.

10 **GetWeekOfYear**

11
12 [C#] public virtual int GetWeekOfYear(DateTime time, CalendarWeekRule rule,
13 DayOfWeek firstDayOfWeek);

14 [C++] public: virtual int GetWeekOfYear(DateTime time, CalendarWeekRule
15 rule, DayOfWeek firstDayOfWeek);

16 [VB] Overridable Public Function GetWeekOfYear(ByVal time As DateTime,
17 ByVal rule As CalendarWeekRule, ByVal firstDayOfWeek As DayOfWeek) As
18 Integer

19 [JScript] public function GetWeekOfYear(time : DateTime, rule :
20 CalendarWeekRule, firstDayOfWeek : DayOfWeek) : int;

21
22 *Description*

23 Gets the week of the year that includes the date in the specified
24 **System.DateTime** .

Return Value: An integer that represents the week of the year that includes the date in *time* .

System.Globalization.DateTimeFormatInfo.FirstDayOfWeek of **System.Globalization.CultureInfo.DateTimeFormat** contains the default **System.DayOfWeek** value that represents the first day of the week for a specific culture. The **System.DateTime** instance to read. A **System.Globalization.CalendarWeekRule** value that defines a calendar week. A **System.DayOfWeek** value that represents the first day of the week.

GetYear

[C#] public abstract int GetYear(DateTime time);

[C++] public: virtual int GetYear(DateTime time) = 0;

[VB] MustOverride Public Function GetYear(ByVal time As DateTime) As

Integer

[JScript] public abstract function GetYear(time : DateTime) : int;

Description

When implemented by a derived class, gets the year in the specified **System.DateTime** .

Return Value: An integer that represents the year in *time* . The **System.DateTime** instance to read.

IsLeapDay

[C#] public virtual bool IsLeapDay(int year, int month, int day);

[C++] public: virtual bool IsLeapDay(int year, int month, int day);

[VB] Overridable Public Function IsLeapDay(ByVal year As Integer, ByVal month As Integer, ByVal day As Integer) As Boolean

[JScript] public function IsLeapDay(year : int, month : int, day : int) : Boolean;

Determines whether a date is a leap day.

Description

Determines whether the date specified by the *year* , *month* , and *day* parameters is a leap day.

Return Value: **true** if the specified day is a leap day; otherwise, **false** .

A leap year has a different number of days than a standard calendar year, in order to make up for the difference between the calendar year and the actual time that the earth rotates around the sun. Each **System.Globalization.Calendar** implementation defines leap years differently. An integer that represents the year. An integer that represents the month. An integer that represents the day.

IsLeapDay

[C#] public abstract bool IsLeapDay(int year, int month, int day, int era);

[C++] public: virtual bool IsLeapDay(int year, int month, int day, int era) = 0;

[VB] MustOverride Public Function IsLeapDay(ByVal year As Integer, ByVal month As Integer, ByVal day As Integer, ByVal era As Integer) As Boolean

[JScript] public abstract function IsLeapDay(year : int, month : int, day : int, era : int) : Boolean;

Description

When implemented by a derived class, determines whether the date specified by the *year* , *month* , *day* , and *era* parameters is a leap day.

Return Value: **true** if the specified day is a leap day; otherwise, **false** . An integer that represents the year. An integer that represents the month. An integer that represents the day. An integer that represents the era.

IsLeapMonth

[C#] public virtual bool IsLeapMonth(int year, int month);

[C++] public: virtual bool IsLeapMonth(int year, int month);

[VB] Overridable Public Function IsLeapMonth(ByVal year As Integer, ByVal month As Integer) As Boolean

[JScript] public function IsLeapMonth(year : int, month : int) : Boolean;

Determines whether a month is a leap month.

Description

Determines whether the month specified by the *year* and *month* parameters is a leap month.

Return Value: **true** if the specified month is a leap month; otherwise, **false** .

A leap year has a different number of days than a standard calendar year, in order to make up for the difference between the calendar year and the actual time that the earth rotates around the sun. Each **System.Globalization.Calendar** implementation defines leap years differently. An integer that represents the year. An integer that represents the month.

IsLeapMonth

```

1
2 [C#] public abstract bool IsLeapMonth(int year, int month, int era);
3 [C++] public: virtual bool IsLeapMonth(int year, int month, int era) = 0;
4 [VB] MustOverride Public Function IsLeapMonth(ByVal year As Integer, ByVal
5 month As Integer, ByVal era As Integer) As Boolean
6 [JScript] public abstract function IsLeapMonth(year : int, month : int, era : int) :
7 Boolean;
8

```

9 *Description*

10 When implemented by a derived class, determines whether the month
11 specified by the *year* , *month* , and *era* parameters is a leap month.

12 *Return Value:* **true** if the specified month is a leap month; otherwise, **false** . An
13 integer that represents the year. An integer that represents the month. An integer
14 that represents the era.

15 *IsLeapYear*

```

16
17 [C#] public virtual bool IsLeapYear(int year);
18 [C++] public: virtual bool IsLeapYear(int year);
19 [VB] Overridable Public Function IsLeapYear(ByVal year As Integer) As
20 Boolean
21 [JScript] public function IsLeapYear(year : int) : Boolean; Determines whether a
22 year is a leap year.
23

```

24 *Description*

Determines whether the year specified by the *year* parameter is a leap year.

Return Value: **true** if the specified year is a leap year; otherwise, **false** .

A leap year has a different number of days than a standard calendar year, in order to make up for the difference between the calendar year and the actual time that the earth rotates around the sun. Each **System.Globalization.Calendar** implementation defines leap years differently. An integer that represents the year.

IsLeapYear

[C#] public abstract bool IsLeapYear(int year, int era);

[C++] public: virtual bool IsLeapYear(int year, int era) = 0;

[VB] MustOverride Public Function IsLeapYear(ByVal year As Integer, ByVal era As Integer) As Boolean

[JScript] public abstract function IsLeapYear(year : int, era : int) : Boolean;

Description

When implemented by a derived class, determines whether the year specified by the *year* and *era* parameters is a leap year.

Return Value: **true** if the specified year is a leap year; otherwise, **false** . An integer that represents the year. An integer that represents the era.

ToDateTime

[C#] public virtual DateTime ToDateTime(int year, int month, int day, int hour, int minute, int second, int millisecond);

[C++] public: virtual DateTime ToDateTime(int year, int month, int day, int hour, int minute, int second, int millisecond);

1 [VB] Overridable Public Function ToDateTime(ByVal year As Integer, ByVal
2 month As Integer, ByVal day As Integer, ByVal hour As Integer, ByVal minute
3 As Integer, ByVal second As Integer, ByVal millisecond As Integer) As DateTime
4 [JScript] public function ToDateTime(year : int, month : int, day : int, hour : int,
5 minute : int, second : int, millisecond : int) : DateTime; Returns a
6 **System.DateTime** that is set to the specified date and time.

8 *Description*

9 Returns a **System.DateTime** that is set to the specified date and time in the
10 current era.

11 *Return Value:* The **System.DateTime** instance set to the specified date and time in
12 the current era. An integer that represents the year. An integer that represents the
13 month. An integer that represents the day. An integer that represents the hour. An
14 integer that represents the minute. An integer that represents the second. An
15 integer that represents the millisecond.

16 ToDateTime

17
18 [C#] public abstract DateTime ToDateTime(int year, int month, int day, int hour,
19 int minute, int second, int millisecond, int era);

20 [C++] public: virtual DateTime ToDateTime(int year, int month, int day, int hour,
21 int minute, int second, int millisecond, int era) = 0;

22 [VB] MustOverride Public Function ToDateTime(ByVal year As Integer, ByVal
23 month As Integer, ByVal day As Integer, ByVal hour As Integer, ByVal minute
24 As Integer, ByVal second As Integer, ByVal millisecond As Integer, ByVal era As
25 Integer) As DateTime

1 [JScript] public abstract function ToDateTime(year : int, month : int, day : int,
2 hour : int, minute : int, second : int, millisecond : int, era : int) : DateTime;

3
4 *Description*

5 When implemented by a derived class, returns a **System.DateTime** that is
6 set to the specified date and time in the specified era.

7 *Return Value:* The **System.DateTime** instance set to the specified date and time in
8 the current era. An integer that represents the year. An integer that represents the
9 month. An integer that represents the day. An integer that represents the hour. An
10 integer that represents the minute. An integer that represents the second. An
11 integer that represents the millisecond. An integer that represents the era.

12 ToFourDigitYear

13
14 [C#] public virtual int ToFourDigitYear(int year);

15 [C++] public: virtual int ToFourDigitYear(int year);

16 [VB] Overridable Public Function ToFourDigitYear(ByVal year As Integer) As
17 Integer

18 [JScript] public function ToFourDigitYear(year : int) : int;

19
20 *Description*

21 Converts the specified two-digit year to a four-digit year by using the
22 **System.Globalization.Calendar.TwoDigitYearMax** property to determine the
23 appropriate century.

24 *Return Value:* An integer that contains the four-digit representation of *year* .
25

System.Globalization.Calendar.TwoDigitYearMax is the last year in the 100-year range that can be represented by a two-digit year. The century is determined by finding the sole occurrence of the two-digit *year* within that 100-year range. For example, if **System.Globalization.Calendar.TwoDigitYearMax** is set to 2029, the 100-year range is from 1930 to 2029; therefore, a 2-digit value of 30 is interpreted as 1930, while a 2-digit value of 29 is interpreted as 2029. A two-digit integer that represents the year to convert.

CalendarWeekRule enumeration (System.Globalization)

ToString

Description

Defines different rules for determining the first week of the year.

These calendar week rules depend on the **System.DayOfWeek** value that is designated as the first day of the week. The

System.Globalization.DateTimeFormatInfo.FirstDayOfWeek property provides the default value for a culture, but any **System.DayOfWeek** value can be specified as the first day of the week in

System.Globalization.Calendar.GetWeekOfYear(System.DateTime, System.Globalization.CalendarWeekRule, System.DayOfWeek) .

ToString

[C#] public const CalendarWeekRule FirstDay;

[C++] public: const CalendarWeekRule FirstDay;

[VB] Public Const FirstDay As CalendarWeekRule

1 [JScript] public var FirstDay : CalendarWeekRule;

3 *Description*

4 Indicates that the first week of the year starts on the first day of the year and
5 ends before the following designated first day of the week. The value is 0.

6 ToString

8 [C#] public const CalendarWeekRule FirstFourDayWeek;

9 [C++] public: const CalendarWeekRule FirstFourDayWeek;

10 [VB] Public Const FirstFourDayWeek As CalendarWeekRule

11 [JScript] public var FirstFourDayWeek : CalendarWeekRule;

13 *Description*

14 Indicates that the first week of the year is the first week with four or more
15 days before the designated first day of the week. The value is 2.

16 ToString

18 [C#] public const CalendarWeekRule FirstFullWeek;

19 [C++] public: const CalendarWeekRule FirstFullWeek;

20 [VB] Public Const FirstFullWeek As CalendarWeekRule

21 [JScript] public var FirstFullWeek : CalendarWeekRule;

23 *Description*

24 Indicates that the first week of the year begins on the first occurrence of the
25 designated first day of the week on or after the first day of the year. The value is 1.

CompareInfo class (System.Globalization)

ToString

Description

Implements a set of methods for culture-sensitive string comparisons.

The **System.Globalization.CultureInfo** class includes a **System.Globalization.CultureInfo.CompareInfo** property that is an instance of this class.

LCID

ToString

[C#] public int LCID {get;}

[C++] public: __property int get_LCID();

[VB] Public ReadOnly Property LCID As Integer

[JScript] public function get LCID() : int;

Description

Gets the properly formed culture identifier for the current **System.Globalization.CompareInfo** instance.

Compare

[C#] public virtual int Compare(string string1, string string2);

[C++] public: virtual int Compare(String* string1, String* string2);

[VB] Overridable Public Function Compare(ByVal string1 As String, ByVal

string2 As String) As Integer

[JScript] public function Compare(string1 : String, string2 : String) : int;

Compares two strings.

Description

Compares two strings using the default

System.Globalization.CompareOptions value.

Return Value: Value Condition zero The two strings are equal. The first string to compare. The second string to compare.

Compare

[C#] public virtual int Compare(string string1, string string2, CompareOptions options);

[C++] public: virtual int Compare(String* string1, String* string2, CompareOptions options);

[VB] Overridable Public Function Compare(ByVal string1 As String, ByVal string2 As String, ByVal options As CompareOptions) As Integer

[JScript] public function Compare(string1 : String, string2 : String, options : CompareOptions) : int;

Description

Compares two strings using the specified

System.Globalization.CompareOptions value.

Return Value: Value Condition zero The two strings are equal. The first string to compare. The second string to compare. The

System.Globalization.CompareOptions value that defines how the strings should be compared.

Compare

```
[C#] public virtual int Compare(string string1, int offset1, string string2, int offset2);
```

```
[C++] public: virtual int Compare(String* string1, int offset1, String* string2, int offset2);
```

```
[VB] Overridable Public Function Compare(ByVal string1 As String, ByVal offset1 As Integer, ByVal string2 As String, ByVal offset2 As Integer) As Integer
```

```
[JScript] public function Compare(string1 : String, offset1 : int, string2 : String, offset2 : int) : int;
```

Description

Compares the tail section of a string with the tail section of another string.

Return Value: Value Condition zero The two strings are equal. The first string to compare. The zero-based index of the character in *string1* at which to start comparing. The second string to compare. The zero-based index of the character in *string2* at which to start comparing.

Compare

```
[C#] public virtual int Compare(string string1, int offset1, string string2, int offset2, CompareOptions options);
```

```
[C++] public: virtual int Compare(String* string1, int offset1, String* string2, int offset2, CompareOptions options);
```

```

1 [VB] Overridable Public Function Compare(ByVal string1 As String, ByVal
2 offset1 As Integer, ByVal string2 As String, ByVal offset2 As Integer, ByVal
3 options As CompareOptions) As Integer
4 [JScript] public function Compare(string1 : String, offset1 : int, string2 : String,
5 offset2 : int, options : CompareOptions) : int;

```

Description

Compares the tail section of a string with the tail section of another string using the specified **System.Globalization.CompareOptions** value.

Return Value: Value Condition zero The two strings are equal. The first string to compare. The zero-based index of the character in *string1* at which to start comparing. The second string to compare. The zero-based index of the character in *string2* at which to start comparing. The **System.Globalization.CompareOptions** value that defines how the strings should be compared.

Compare

```

17 [C#] public virtual int Compare(string string1, int offset1, int length1, string
18 string2, int offset2, int length2);
19 [C++] public: virtual int Compare(String* string1, int offset1, int length1, String*
20 string2, int offset2, int length2);
21 [VB] Overridable Public Function Compare(ByVal string1 As String, ByVal
22 offset1 As Integer, ByVal length1 As Integer, ByVal string2 As String, ByVal
23 offset2 As Integer, ByVal length2 As Integer) As Integer
24 [JScript] public function Compare(string1 : String, offset1 : int, length1 : int,
25 string2 : String, offset2 : int, length2 : int) : int;

```

Description

Compares a section of one string with a section of another string.

Return Value: Value Condition zero The two strings are equal. The first string to compare. The zero-based index of the character in *string1* at which to start comparing. The number of consecutive characters in *string1* to compare. The second string to compare. The zero-based index of the character in *string2* at which to start comparing. The number of consecutive characters in *string2* to compare.

Compare

```
[C#] public virtual int Compare(string string1, int offset1, int length1, string  
string2, int offset2, int length2, CompareOptions options);
```

```
[C++] public: virtual int Compare(String* string1, int offset1, int length1, String*  
string2, int offset2, int length2, CompareOptions options);
```

```
[VB] Overridable Public Function Compare(ByVal string1 As String, ByVal  
offset1 As Integer, ByVal length1 As Integer, ByVal string2 As String, ByVal  
offset2 As Integer, ByVal length2 As Integer, ByVal options As CompareOptions)  
As Integer
```

```
[JScript] public function Compare(string1 : String, offset1 : int, length1 : int,  
string2 : String, offset2 : int, length2 : int, options : CompareOptions) : int;
```

Description

Compares a section of one string with a section of another string using the specified **System.Globalization.CompareOptions** value.

Return Value: Value Condition zero The two strings are equal. The first string to compare. The zero-based index of the character in *string1* at which to start comparing. The number of consecutive characters in *string1* to compare. The second string to compare. The zero-based index of the character in *string2* at which to start comparing. The number of consecutive characters in *string2* to compare. The **System.Globalization.CompareOptions** value that defines how the strings should be compared.

Equals

[C#] public override bool Equals(object value);

[C++] public: bool Equals(Object* value);

[VB] Overrides Public Function Equals(ByVal value As Object) As Boolean

[JScript] public override function Equals(value : Object) : Boolean;

Description

Determines whether the specified **System.Object** is the same instance as the current **System.Globalization.CompareInfo** .

Return Value: **true** if the specified **System.Object** is the same instance as the current **System.Globalization.CompareInfo** ; otherwise, **false** .

This method overrides **System.Object.Equals(System.Object)** . The **System.Object** to compare with the current **System.Globalization.CompareInfo**.

GetCompareInfo

[C#] public static CompareInfo GetCompareInfo(int culture);

[C++] public: static CompareInfo* GetCompareInfo(int culture);

1 [VB] Public Shared Function GetCompareInfo(ByVal culture As Integer) As
2 CompareInfo

3 [JScript] public static function GetCompareInfo(culture : int) : CompareInfo;

4
5 *Description*

6 Initializes a new instance of the **System.Globalization.CompareInfo** class
7 that is associated with the culture having the specified identifier.

8 *Return Value:* A new instance of the **System.Globalization.CompareInfo** class
9 that is associated with the culture having the specified identifier and uses string
10 comparison methods in the current **System.Reflection.Assembly** . An integer
11 representing the culture identifier.

12 GetCompareInfo

13
14 [C#] public static CompareInfo GetCompareInfo(string name);

15 [C++] public: static CompareInfo* GetCompareInfo(String* name);

16 [VB] Public Shared Function GetCompareInfo(ByVal name As String) As
17 CompareInfo

18 [JScript] public static function GetCompareInfo(name : String) : CompareInfo;

19
20 *Description*

21 Initializes a new instance of the **System.Globalization.CompareInfo** class
22 that is associated with the culture having the specified name.

23 *Return Value:* A new instance of the **System.Globalization.CompareInfo** class
24 that is associated with the culture having the specified name and uses string
25

comparison methods in the current **System.Reflection.Assembly** . A

System.String representing the culture name.

GetCompareInfo

[C#] public static CompareInfo GetCompareInfo(int culture, Assembly assembly);

[C++] public: static CompareInfo* GetCompareInfo(int culture, Assembly* assembly);

[VB] Public Shared Function GetCompareInfo(ByVal culture As Integer, ByVal assembly As Assembly) As CompareInfo

[JScript] public static function GetCompareInfo(culture : int, assembly :

Assembly) : CompareInfo; Initializes a new instance of the

System.Globalization.CompareInfo class.

Description

Initializes a new instance of the **System.Globalization.CompareInfo** class that is associated with the culture having the specified identifier and uses string comparison methods in the specified **System.Reflection.Assembly** .

Return Value: A new instance of the **System.Globalization.CompareInfo** class that is associated with the specified culture and uses string comparison methods in the specified **System.Reflection.Assembly** .

The assembly parameter must be of the same type as **System.Reflection.Module.Assembly** . An integer representing the culture identifier. An **System.Reflection.Assembly** that contains the string comparison methods to use.

GetCompareInfo


```

1
2 [C#] public static CompareInfo GetCompareInfo(string name, Assembly
3 assembly);
4 [C++] public: static CompareInfo* GetCompareInfo(String* name, Assembly*
5 assembly);
6 [VB] Public Shared Function GetCompareInfo(ByVal name As String, ByVal
7 assembly As Assembly) As CompareInfo
8 [JScript] public static function GetCompareInfo(name : String, assembly :
9 Assembly) : CompareInfo;
10

```

Description

Initializes a new instance of the **System.Globalization.CompareInfo** class that is associated with the culture having the specified name and uses string comparison methods in the specified **System.Reflection.Assembly** .

Return Value: A new instance of the **System.Globalization.CompareInfo** class that is associated with the culture having the specified name and uses string comparison methods in the specified **System.Reflection.Assembly** .

The assembly parameter must be of the same type as **System.Reflection.Module.Assembly** . A **System.String** representing the culture name. An **System.Reflection.Assembly** that contains the string comparison methods to use.

GetHashCode

```

23
24 [C#] public override int GetHashCode();
25 [C++] public: int GetHashCode();

```

1 [VB] Overrides Public Function GetHashCode() As Integer

2 [JScript] public override function GetHashCode() : int;

3
4 *Description*

5 Serves as a hash function for the current

6 **System.Globalization.CompareInfo** instance, suitable for use in hashing
7 algorithms and data structures, such as a hash table.

8 *Return Value:* A hash code for the current **System.Globalization.CompareInfo**
9 instance.

10 This method overrides **System.Object.GetHashCode** .

11 **GetSortKey**

12
13 [C#] public virtual SortKey GetSortKey(string source);

14 [C++] public: virtual SortKey* GetSortKey(String* source);

15 [VB] Overridable Public Function GetSortKey(ByVal source As String) As
16 SortKey

17 [JScript] public function GetSortKey(source : String) : SortKey;

18
19 *Description*

20 Gets the **System.Globalization.SortKey** of the specified **System.String** .

21 *Return Value:* The **System.Globalization.SortKey** of the specified **System.String**

22 .

23 Each character in a string is given several categories of sort weights,
24 including script, alphabetic, case, and diacritic weights. A sort key serves as the
25 repository of these weights for a particular string. For example, a sort key might

contain a string of alphabetic weights, followed by a string of case weights, and so on. The **System.String** for which to get the **System.Globalization.SortKey**.

GetSortKey

[C#] public virtual SortKey GetSortKey(string source, CompareOptions options);

[C++] public: virtual SortKey* GetSortKey(String* source, CompareOptions options);

[VB] Overridable Public Function GetSortKey(ByVal source As String, ByVal options As CompareOptions) As SortKey

[JScript] public function GetSortKey(source : String, options : CompareOptions) : SortKey; Gets the **System.Globalization.SortKey** of a **System.String**.

Description

Gets the **System.Globalization.SortKey** of the specified **System.String** using the specified **System.Globalization.CompareOptions** value.

Return Value: The **System.Globalization.SortKey** of the specified **System.String** using the specified **System.Globalization.CompareOptions** value.

Each character in a string is given several categories of sort weights, including script, alphabetic, case, and diacritic weights. A sort key serves as the repository of these weights for a particular string. For example, a sort key might contain a string of alphabetic weights, followed by a string of case weights, and so on. The **System.String** whose **System.Globalization.SortKey** to get. The **System.Globalization.CompareOptions** value that defines how the strings should be compared.

IndexOf

1 [C#] public virtual int IndexOf(string source, char value);
 2 [C++] public: virtual int IndexOf(String* source, __wchar_t value);
 3 [VB] Overridable Public Function IndexOf(ByVal source As String, ByVal value
 4 As Char) As Integer
 5 [JScript] public function IndexOf(source : String, value : Char) : int; Returns the
 6 zero-based index of the first occurrence of a value within a **System.String** or
 7 within a portion of it.

10 *Description*

11 Searches for the specified character and returns the zero-based index of the
 12 first occurrence within the entire source **System.String** .

13 *Return Value:* The zero-based index of the first occurrence of *value* within the
 14 entire *source* , if found; otherwise, -1.

15 The source **System.String** is searched forward starting at the beginning of
 16 the **System.String** and ending at the end of the **System.String** . The
 17 **System.String** to search. The character to locate within *source*.

18 **IndexOf**

19
 20 [C#] public virtual int IndexOf(string source, string value);
 21 [C++] public: virtual int IndexOf(String* source, String* value);
 22 [VB] Overridable Public Function IndexOf(ByVal source As String, ByVal value
 23 As String) As Integer
 24 [JScript] public function IndexOf(source : String, value : String) : int;
 25

Description

Searches for the specified substring and returns the zero-based index of the first occurrence within the entire source **System.String**.

Return Value: The zero-based index of the first occurrence of *value* within the entire *source*, if found; otherwise, -1.

The source **System.String** is searched forward starting at the beginning of the **System.String** and ending at the end of the **System.String**. The **System.String** to search. The **System.String** to locate within *source*.

IndexOf

[C#] public virtual int IndexOf(string source, char value, CompareOptions options);

[C++] public: virtual int IndexOf(String* source, __wchar_t value, CompareOptions options);

[VB] Overridable Public Function IndexOf(ByVal source As String, ByVal value As Char, ByVal options As CompareOptions) As Integer

[JScript] public function IndexOf(source : String, value : Char, options : CompareOptions) : int;

Description

Searches for the specified character and returns the zero-based index of the first occurrence within the entire source **System.String** using the specified **System.Globalization.CompareOptions** value.

Return Value: The zero-based index of the first occurrence of *value* within the

entire *source* using the specified **System.Globalization.CompareOptions** value, if found; otherwise, -1.

The source **System.String** is searched forward starting at the beginning of the **System.String** and ending at the end of the **System.String** . The **System.String** to search. The character to locate within *source*. The **System.Globalization.CompareOptions** value that defines how the strings should be compared.

IndexOf

[C#] public virtual int IndexOf(string source, char value, int startIndex);

[C++] public: virtual int IndexOf(String* source, __wchar_t value, int startIndex);

[VB] Overridable Public Function IndexOf(ByVal source As String, ByVal value As Char, ByVal startIndex As Integer) As Integer

[JScript] public function IndexOf(source : String, value : Char, startIndex : int) : int;

Description

Searches for the specified character and returns the zero-based index of the first occurrence within the section of the source **System.String** that extends from the specified index to the end of the **System.String** .

Return Value: The zero-based index of the first occurrence of *value* within the section of *source* that extends from *startIndex* to the end of the **System.String** , if found; otherwise, -1.

1 The source **System.String** is searched forward starting at *startIndex* and
2 ending at the end of the **System.String** . The **System.String** to search. The
3 character to locate within *source*. The zero-based starting index of the search.

4 IndexOf

5
6 [C#] public virtual int IndexOf(string source, string value, CompareOptions
7 options);

8 [C++] public: virtual int IndexOf(String* source, String* value, CompareOptions
9 options);

10 [VB] Overridable Public Function IndexOf(ByVal source As String, ByVal value
11 As String, ByVal options As CompareOptions) As Integer

12 [JScript] public function IndexOf(source : String, value : String, options :
13 CompareOptions) : int;

15 Description

16 Searches for the specified substring and returns the zero-based index of the
17 first occurrence within the entire source **System.String** using the specified
18 **System.Globalization.CompareOptions** value.

19 *Return Value:* The zero-based index of the first occurrence of *value* within the
20 entire *source* using the specified **System.Globalization.CompareOptions** value,
21 if found; otherwise, -1.

22 The source **System.String** is searched forward starting at the beginning of
23 the **System.String** and ending at the end of the **System.String** . The
24 **System.String** to search. The **System.String** to locate within *source*. The
25

System.Globalization.CompareOptions value that defines how the strings should be compared.

IndexOf

[C#] public virtual int IndexOf(string source, string value, int startIndex);

[C++] public: virtual int IndexOf(String* source, String* value, int startIndex);

[VB] Overridable Public Function IndexOf(ByVal source As String, ByVal value As String, ByVal startIndex As Integer) As Integer

[JScript] public function IndexOf(source : String, value : String, startIndex : int) : int;

Description

Searches for the specified substring and returns the zero-based index of the first occurrence within the section of the source **System.String** that extends from the specified index to the end of the **System.String**.

Return Value: The zero-based index of the first occurrence of *value* within the section of *source* that extends from *startIndex* to the end of the **System.String**, if found; otherwise, -1.

The source **System.String** is searched forward starting at *startIndex* and ending at the end of the **System.String**. The **System.String** to search. The **System.String** to locate within *source*. The zero-based starting index of the search.

IndexOf

[C#] public virtual int IndexOf(string source, char value, int startIndex,

1 CompareOptions options);

2 [C++] public: virtual int IndexOf(String* source, __wchar_t value, int startIndex,

3 CompareOptions options);

4 [VB] Overridable Public Function IndexOf(ByVal source As String, ByVal value

5 As Char, ByVal startIndex As Integer, ByVal options As CompareOptions) As

6 Integer

7 [JScript] public function IndexOf(source : String, value : Char, startIndex : int,

8 options : CompareOptions) : int;

9 *Description*

10
11 Searches for the specified character and returns the zero-based index of the
12 first occurrence within the section of the source **System.String** that extends from
13 the specified index to the end of the **System.String** using the specified
14 **System.Globalization.CompareOptions** value.

15 *Return Value:* The zero-based index of the first occurrence of *value* within the
16 section of *source* that extends from *startIndex* to the end of the **System.String**
17 using the specified **System.Globalization.CompareOptions** value, if found;
18 otherwise, -1.

19 The source **System.String** is searched forward starting at *startIndex* and
20 ending at the end of the **System.String**. The **System.String** to search. The
21 character to locate within *source*. The zero-based starting index of the search. The
22 **System.Globalization.CompareOptions** value that defines how the strings
23 should be compared.

24 IndexOf

```

1
2 [C#] public virtual int IndexOf(string source, char value, int startIndex, int count);
3 [C++] public: virtual int IndexOf(String* source, __wchar_t value, int startIndex,
4 int count);
5 [VB] Overridable Public Function IndexOf(ByVal source As String, ByVal value
6 As Char, ByVal startIndex As Integer, ByVal count As Integer) As Integer
7 [JScript] public function IndexOf(source : String, value : Char, startIndex : int,
8 count : int) : int;
9

```

Description

Searches for the specified character and returns the zero-based index of the first occurrence within the section of the source **System.String** that starts at the specified index and contains the specified number of elements.

Return Value: The zero-based index of the first occurrence of *value* within the section of *source* that starts at *startIndex* and contains *count* number of elements, if found; otherwise, -1.

The source **System.String** is searched forward starting at *startIndex* and ending at *startIndex + count - 1*. The **System.String** to search. The character to locate within *source*. The zero-based starting index of the search. The number of elements in the section to search.

IndexOf

```

22
23 [C#] public virtual int IndexOf(string source, string value, int startIndex,
24 CompareOptions options);
25 [C++] public: virtual int IndexOf(String* source, String* value, int startIndex,

```

1 CompareOptions options);

2 [VB] Overridable Public Function IndexOf(ByVal source As String, ByVal value
3 As String, ByVal startIndex As Integer, ByVal options As CompareOptions) As
4 Integer

5 [JScript] public function IndexOf(source : String, value : String, startIndex : int,
6 options : CompareOptions) : int;

8 *Description*

9 Searches for the specified substring and returns the zero-based index of the
10 first occurrence within the section of the source **System.String** that extends from
11 the specified index to the end of the **System.String** using the specified
12 **System.Globalization.CompareOptions** value.

13 *Return Value:* The zero-based index of the first occurrence of *value* within the
14 section of *source* that extends from *startIndex* to the end of the **System.String**
15 using the specified **System.Globalization.CompareOptions** value, if found;
16 otherwise, -1.

17 The source **System.String** is searched forward starting at *startIndex* and
18 ending at the end of the **System.String**. The **System.String** to search. The
19 **System.String** to locate within *source*. The zero-based starting index of the
20 search. The **System.Globalization.CompareOptions** value that defines how the
21 strings should be compared.

22 *IndexOf*

23
24 [C#] public virtual int IndexOf(string source, string value, int startIndex, int
25 count);

1 [C++] public: virtual int IndexOf(String* source, String* value, int startIndex, int
2 count);

3 [VB] Overridable Public Function IndexOf(ByVal source As String, ByVal value
4 As String, ByVal startIndex As Integer, ByVal count As Integer) As Integer

5 [JScript] public function IndexOf(source : String, value : String, startIndex : int,
6 count : int) : int;

8 *Description*

9 Searches for the specified substring and returns the zero-based index of the
10 first occurrence within the section of the source **System.String** that starts at the
11 specified index and contains the specified number of elements.

12 *Return Value:* The zero-based index of the first occurrence of *value* within the
13 section of *source* that starts at *startIndex* and contains *count* number of elements,
14 if found; otherwise, -1.

15 The source **System.String** is searched forward starting at *startIndex* and
16 ending at *startIndex* + *count* - 1. The **System.String** to search. The **System.String**
17 to locate within *source*. The zero-based starting index of the search. The number
18 of elements in the section to search.

19 **IndexOf**

20
21 [C#] public virtual int IndexOf(string source, char value, int startIndex, int count,
22 CompareOptions options);

23 [C++] public: virtual int IndexOf(String* source, __wchar_t value, int startIndex,
24 int count, CompareOptions options);

25 [VB] Overridable Public Function IndexOf(ByVal source As String, ByVal value

As Char, ByVal startIndex As Integer, ByVal count As Integer, ByVal options As
CompareOptions) As Integer
[JScript] public function IndexOf(source : String, value : Char, startIndex : int,
count : int, options : CompareOptions) : int;

Description

Searches for the specified character and returns the zero-based index of the first occurrence within the section of the source **System.String** that starts at the specified index and contains the specified number of elements using the specified **System.Globalization.CompareOptions** value.

Return Value: The zero-based index of the first occurrence of *value* within the section of *source* that starts at *startIndex* and contains *count* number of elements using the specified **System.Globalization.CompareOptions** value, if found; otherwise, -1.

The source **System.String** is searched forward starting at *startIndex* and ending at *startIndex* + *count* - 1. The **System.String** to search. The character to locate within *source*. The zero-based starting index of the search. The number of elements in the section to search. The **System.Globalization.CompareOptions** value that defines how the strings should be compared.

IndexOf

[C#] public virtual int IndexOf(string source, string value, int startIndex, int count, CompareOptions options);
[C++] public: virtual int IndexOf(String* source, String* value, int startIndex, int count, CompareOptions options);

1 [VB] Overridable Public Function IndexOf(ByVal source As String, ByVal value
2 As String, ByVal startIndex As Integer, ByVal count As Integer, ByVal options
3 As CompareOptions) As Integer

4 [JScript] public function IndexOf(source : String, value : String, startIndex : int,
5 count : int, options : CompareOptions) : int;

6 7 *Description*

8 Searches for the specified substring and returns the zero-based index of the
9 first occurrence within the section of the source **System.String** that starts at the
10 specified index and contains the specified number of elements using the specified
11 **System.Globalization.CompareOptions** value.

12 *Return Value:* The zero-based index of the first occurrence of *value* within the
13 section of *source* that starts at *startIndex* and contains *count* number of elements
14 using the specified **System.Globalization.CompareOptions** value, if found;
15 otherwise, -1.

16 The source **System.String** is searched forward starting at *startIndex* and
17 ending at *startIndex* + *count* - 1. The **System.String** to search. The **System.String**
18 to locate within *source*. The zero-based starting index of the search. The number
19 of elements in the section to search. The **System.Globalization.CompareOptions**
20 value that defines how the strings should be compared.

21 **IsPrefix**

22
23 [C#] public virtual bool IsPrefix(string source, string prefix);

24 [C++] public: virtual bool IsPrefix(String* source, String* prefix);

25 [VB] Overridable Public Function IsPrefix(ByVal source As String, ByVal prefix

As String) As Boolean

[JScript] public function IsPrefix(source : String, prefix : String) : Boolean;

Description

Determines whether the specified source **System.String** starts with the specified prefix.

Return Value: **true** if the length of *prefix* is less than or equal to the length of *source* and *source* starts with *prefix* ; otherwise, **false** .

Every string starts and ends with an empty substring; therefore, if *prefix* is an empty string, this method returns **true** . The **System.String** to search in. The **System.String** to compare with the beginning of *source*.

IsPrefix

[C#] public virtual bool IsPrefix(string source, string prefix, CompareOptions options);

[C++] public: virtual bool IsPrefix(String* source, String* prefix, CompareOptions options);

[VB] Overridable Public Function IsPrefix(ByVal source As String, ByVal prefix As String, ByVal options As CompareOptions) As Boolean

[JScript] public function IsPrefix(source : String, prefix : String, options : CompareOptions) : Boolean; Determines whether a string starts with a specific prefix.

Description

Determines whether the specified source **System.String** starts with the specified prefix using the specified **System.Globalization.CompareOptions** value.

Return Value: **true** if the length of *prefix* is less than or equal to the length of *source* and *source* starts with *prefix* ; otherwise, **false** .

Every string starts and ends with an empty substring; therefore, if *prefix* is an empty string, this method returns **true** . The **System.String** to search in. The **System.String** to compare with the beginning of *source*. The **System.Globalization.CompareOptions** value that defines how the strings should be compared.

IsSuffix

[C#] public virtual bool IsSuffix(string source, string suffix);

[C++] public: virtual bool IsSuffix(String* source, String* suffix);

[VB] Overridable Public Function IsSuffix(ByVal source As String, ByVal suffix As String) As Boolean

[JScript] public function IsSuffix(source : String, suffix : String) : Boolean;

Description

Determines whether the specified source **System.String** ends with the specified suffix.

Return Value: **true** if the length of *suffix* is less than or equal to the length of *source* and *source* ends with *suffix* ; otherwise, **false** .

Every string starts and ends with an empty substring; therefore, if *suffix* is an empty string, this method returns **true** . The **System.String** to search in. The **System.String** to compare with the end of *source*.

IsSuffix

[C#] public virtual bool IsSuffix(string source, string suffix, CompareOptions options);

[C++] public: virtual bool IsSuffix(String* source, String* suffix, CompareOptions options);

[VB] Overridable Public Function IsSuffix(ByVal source As String, ByVal suffix As String, ByVal options As CompareOptions) As Boolean

[JScript] public function IsSuffix(source : String, suffix : String, options : CompareOptions) : Boolean; Determines whether a string ends with a specific suffix.

Description

Determines whether the specified source **System.String** ends with the specified suffix using the specified **System.Globalization.CompareOptions** value.

Return Value: **true** if the length of *suffix* is less than or equal to the length of *source* and *source* ends with *suffix* ; otherwise, **false** .

Every string starts and ends with an empty substring; therefore, if *suffix* is an empty string, this method returns **true** . The **System.String** to search in. The **System.String** to compare with the end of *source*. The

System.Globalization.CompareOptions value that defines how the strings should be compared.

LastIndexOf

[C#] public virtual int LastIndexOf(string source, char value);

[C++] public: virtual int LastIndexOf(String* source, __wchar_t value);

[VB] Overridable Public Function LastIndexOf(ByVal source As String, ByVal value As Char) As Integer

[JScript] public function LastIndexOf(source : String, value : Char) : int; Returns the zero-based index of the last occurrence of a value within a **System.String** or within a portion of it.

Description

Searches for the specified character and returns the zero-based index of the last occurrence within the entire source **System.String**.

Return Value: The zero-based index of the last occurrence of *value* within the entire *source*, if found; otherwise, -1.

The source **System.String** is searched backward starting at the end of the **System.String** and ending at the beginning of the **System.String**. The **System.String** to search. The character to locate within *source*.

LastIndexOf

[C#] public virtual int LastIndexOf(string source, string value);

[C++] public: virtual int LastIndexOf(String* source, String* value);

[VB] Overridable Public Function LastIndexOf(ByVal source As String, ByVal

1 value As String) As Integer

2 [JScript] public function LastIndexOf(source : String, value : String) : int;

3
4 *Description*

5 Searches for the specified substring and returns the zero-based index of the
6 last occurrence within the entire source **System.String** .

7 *Return Value:* The zero-based index of the last occurrence of *value* within the
8 entire *source* , if found; otherwise, -1.

9 The source **System.String** is searched backward starting at the end of the
10 **System.String** and ending at the beginning of the **System.String** . The
11 **System.String** to search. The **System.String** to locate within *source*.

12 LastIndexOf

13
14 [C#] public virtual int LastIndexOf(string source, char value, CompareOptions
15 options);

16 [C++] public: virtual int LastIndexOf(String* source, __wchar_t value,
17 CompareOptions options);

18 [VB] Overridable Public Function LastIndexOf(ByVal source As String, ByVal
19 value As Char, ByVal options As CompareOptions) As Integer

20 [JScript] public function LastIndexOf(source : String, value : Char, options :
21 CompareOptions) : int;

22
23 *Description*

24 Searches for the specified character and returns the zero-based index of the
25 last occurrence within the entire source **System.String** using the specified

System.Globalization.CompareOptions value.

Return Value: The zero-based index of the last occurrence of *value* within the entire *source* using the specified **System.Globalization.CompareOptions** value, if found; otherwise, -1.

The source **System.String** is searched backward starting at the end of the **System.String** and ending at the beginning of the **System.String**. The **System.String** to search. The character to locate within *source*. The **System.Globalization.CompareOptions** value that defines how the strings should be compared.

LastIndexOf

[C#] public virtual int LastIndexOf(string source, char value, int startIndex);

[C++] public: virtual int LastIndexOf(String* source, __wchar_t value, int startIndex);

[VB] Overridable Public Function LastIndexOf(ByVal source As String, ByVal value As Char, ByVal startIndex As Integer) As Integer

[JScript] public function LastIndexOf(source : String, value : Char, startIndex : int) : int;

Description

Searches for the specified character and returns the zero-based index of the last occurrence within the section of the source **System.String** that extends from the beginning of the **System.String** to the specified index.

Return Value: The zero-based index of the last occurrence of *value* within the

1 section of *source* that extends from the beginning of the **System.String** to
2 *startIndex* , if found; otherwise, -1.

3 The source **System.String** is searched backward starting at *startIndex* and
4 ending at the beginning of the **System.String** . The **System.String** to search. The
5 character to locate within *source*. The zero-based starting index of the backward
6 search.

7 LastIndexOf

8
9 [C#] public virtual int LastIndexOf(string source, string value, CompareOptions
10 options);

11 [C++] public: virtual int LastIndexOf(String* source, String* value,
12 CompareOptions options);

13 [VB] Overridable Public Function LastIndexOf(ByVal source As String, ByVal
14 value As String, ByVal options As CompareOptions) As Integer

15 [JScript] public function LastIndexOf(source : String, value : String, options :
16 CompareOptions) : int;

17 Description

18
19 Searches for the specified substring and returns the zero-based index of the
20 last occurrence within the entire source **System.String** using the specified
21 **System.Globalization.CompareOptions** value.

22 *Return Value:* The zero-based index of the last occurrence of *value* within the
23 entire *source* using the specified **System.Globalization.CompareOptions** value,
24 if found; otherwise, -1.
25

1 The source **System.String** is searched backward starting at the end of the
2 **System.String** and ending at the beginning of the **System.String** . The
3 **System.String** to search. The **System.String** to locate within *source*. The
4 **System.Globalization.CompareOptions** value that defines how the strings
5 should be compared.

6 LastIndexOf

7
8 [C#] public virtual int LastIndexOf(string source, string value, int startIndex);

9 [C++] public: virtual int LastIndexOf(String* source, String* value, int
10 startIndex);

11 [VB] Overridable Public Function LastIndexOf(ByVal source As String, ByVal
12 value As String, ByVal startIndex As Integer) As Integer

13 [JScript] public function LastIndexOf(source : String, value : String, startIndex :
14 int) : int;

15 Description

16
17 Searches for the specified substring and returns the zero-based index of the
18 last occurrence within the section of the source **System.String** that extends from
19 the beginning of the **System.String** to the specified index.

20 *Return Value:* The zero-based index of the last occurrence of *value* within the
21 section of *source* that extends from the beginning of the **System.String** to
22 *startIndex* , if found; otherwise, -1.

23 The source **System.String** is searched backward starting at *startIndex* and
24 ending at the beginning of the **System.String** . The **System.String** to search. The
25

System.String to locate within *source*. The zero-based starting index of the backward search.

LastIndexOf

[C#] public virtual int LastIndexOf(string source, char value, int startIndex, CompareOptions options);

[C++] public: virtual int LastIndexOf(String* source, __wchar_t value, int startIndex, CompareOptions options);

[VB] Overridable Public Function LastIndexOf(ByVal source As String, ByVal value As Char, ByVal startIndex As Integer, ByVal options As CompareOptions) As Integer

[JScript] public function LastIndexOf(source : String, value : Char, startIndex : int, options : CompareOptions) : int;

Description

Searches for the specified character and returns the zero-based index of the last occurrence within the section of the source **System.String** that extends from the beginning of the **System.String** to the specified index using the specified **System.Globalization.CompareOptions** value.

Return Value: The zero-based index of the last occurrence of *value* within the section of *source* that extends from the beginning of the **System.String** to *startIndex* using the specified **System.Globalization.CompareOptions** value, if found; otherwise, -1.

The source **System.String** is searched backward starting at *startIndex* and ending at the beginning of the **System.String**. The **System.String** to search. The

character to locate within *source*. The zero-based starting index of the backward search. The **System.Globalization.CompareOptions** value that defines how the strings should be compared.

LastIndexOf

[C#] public virtual int LastIndexOf(string source, char value, int startIndex, int count);

[C++] public: virtual int LastIndexOf(String* source, __wchar_t value, int startIndex, int count);

[VB] Overridable Public Function LastIndexOf(ByVal source As String, ByVal value As Char, ByVal startIndex As Integer, ByVal count As Integer) As Integer

[JScript] public function LastIndexOf(source : String, value : Char, startIndex : int, count : int) : int;

Description

Searches for the specified character and returns the zero-based index of the last occurrence within the section of the source **System.String** that contains the specified number of elements and ends at the specified index.

Return Value: The zero-based index of the last occurrence of *value* within the section of *source* that contains *count* number of elements and ends at *startIndex* , if found; otherwise, -1.

The source **System.String** is searched backward starting at *startIndex* and ending at *startIndex - count + 1*. The **System.String** to search. The character to locate within *source*. The zero-based starting index of the backward search. The number of elements in the section to search.

LastIndexOf

```
[C#] public virtual int LastIndexOf(string source, string value, int startIndex,  
CompareOptions options);  
[C++] public: virtual int LastIndexOf(String* source, String* value, int startIndex,  
CompareOptions options);  
[VB] Overridable Public Function LastIndexOf(ByVal source As String, ByVal  
value As String, ByVal startIndex As Integer, ByVal options As CompareOptions)  
As Integer  
[JScript] public function LastIndexOf(source : String, value : String, startIndex :  
int, options : CompareOptions) : int;
```

Description

Searches for the specified substring and returns the zero-based index of the last occurrence within the section of the source **System.String** that extends from the beginning of the **System.String** to the specified index using the specified **System.Globalization.CompareOptions** value.

Return Value: The zero-based index of the last occurrence of *value* within the section of *source* that extends from the beginning of the **System.String** to *startIndex* using the specified **System.Globalization.CompareOptions** value, if found; otherwise, -1.

The source **System.String** is searched backward starting at *startIndex* and ending at the beginning of the **System.String**. The **System.String** to search. The **System.String** to locate within *source*. The zero-based starting index of the

backward search. The **System.Globalization.CompareOptions** value that defines how the strings should be compared.

LastIndexOf

[C#] public virtual int LastIndexOf(string source, string value, int startIndex, int count);

[C++] public: virtual int LastIndexOf(String* source, String* value, int startIndex, int count);

[VB] Overridable Public Function LastIndexOf(ByVal source As String, ByVal value As String, ByVal startIndex As Integer, ByVal count As Integer) As Integer

[JScript] public function LastIndexOf(source : String, value : String, startIndex : int, count : int) : int;

Description

Searches for the specified substring and returns the zero-based index of the last occurrence within the section of the source **System.String** that contains the specified number of elements and ends at the specified index.

Return Value: The zero-based index of the last occurrence of *value* within the section of *source* that contains *count* number of elements and ends at *startIndex* , if found; otherwise, -1.

The source **System.String** is searched backward starting at *startIndex* and ending at *startIndex - count + 1*. The **System.String** to search. The **System.String** to locate within *source*. The zero-based starting index of the backward search. The number of elements in the section to search.

LastIndexOf

```

1
2 [C#] public virtual int LastIndexOf(string source, char value, int startIndex, int
3 count, CompareOptions options);
4 [C++] public: virtual int LastIndexOf(String* source, __wchar_t value, int
5 startIndex, int count, CompareOptions options);
6 [VB] Overridable Public Function LastIndexOf(ByVal source As String, ByVal
7 value As Char, ByVal startIndex As Integer, ByVal count As Integer, ByVal
8 options As CompareOptions) As Integer
9 [JScript] public function LastIndexOf(source : String, value : Char, startIndex : int,
10 count : int, options : CompareOptions) : int;
11

```

Description

Searches for the specified character and returns the zero-based index of the last occurrence within the section of the source **System.String** that contains the specified number of elements and ends at the specified index using the specified **System.Globalization.CompareOptions** value.

Return Value: The zero-based index of the last occurrence of *value* within the section of *source* that contains *count* number of elements and ends at *startIndex* using the specified **System.Globalization.CompareOptions** value, if found; otherwise, -1.

The source **System.String** is searched backward starting at *startIndex* and ending at *startIndex - count + 1*. The **System.String** to search. The character to locate within *source*. The zero-based starting index of the backward search. The number of elements in the section to search. The

System.Globalization.CompareOptions value that defines how the strings should be compared.

LastIndexOf

[C#] public virtual int LastIndexOf(string source, string value, int startIndex, int count, CompareOptions options);

[C++] public: virtual int LastIndexOf(String* source, String* value, int startIndex, int count, CompareOptions options);

[VB] Overridable Public Function LastIndexOf(ByVal source As String, ByVal value As String, ByVal startIndex As Integer, ByVal count As Integer, ByVal options As CompareOptions) As Integer

[JScript] public function LastIndexOf(source : String, value : String, startIndex : int, count : int, options : CompareOptions) : int;

Description

Searches for the specified substring and returns the zero-based index of the last occurrence within the section of the source **System.String** that contains the specified number of elements and ends at the specified index using the specified **System.Globalization.CompareOptions** value.

Return Value: The zero-based index of the last occurrence of *value* within the section of *source* that contains *count* number of elements and ends at *startIndex* using the specified **System.Globalization.CompareOptions** value, if found; otherwise, -1.

The source **System.String** is searched backward starting at *startIndex* and ending at *startIndex - count + 1*. The **System.String** to search. The **System.String**

to locate within *source*. The zero-based starting index of the backward search. The number of elements in the section to search. The **System.Globalization.CompareOptions** value that defines how the strings should be compared.

IDeserializationCallback.OnDeserialization

[C#] void IDeserializationCallback.OnDeserialization(object sender);

[C++] void IDeserializationCallback::OnDeserialization(Object* sender);

[VB] Sub OnDeserialization(ByVal sender As Object) Implements

IDeserializationCallback.OnDeserialization

[JScript] function IDeserializationCallback.OnDeserialization(sender : Object);

ToString

[C#] public override string ToString();

[C++] public: String* ToString();

[VB] Overrides Public Function ToString() As String

[JScript] public override function ToString() : String;

Description

Returns a **System.String** that represents the current **System.Globalization.CompareInfo** instance.

Return Value: A **System.String** that represents the current **System.Globalization.CompareInfo** instance.

This method overrides **System.Object.ToString**.

CompareOptions enumeration (System.Globalization)

ToString

Description

Defines the string comparison options to use with

System.Globalization.CompareInfo .

These options denote case-sensitivity or whether to ignore types of characters.

ToString

[C#] public const CompareOptions IgnoreCase;

[C++] public: const CompareOptions IgnoreCase;

[VB] Public Const IgnoreCase As CompareOptions

[JScript] public var IgnoreCase : CompareOptions;

Description

Indicates that the string comparison must ignore case.

ToString

[C#] public const CompareOptions IgnoreKanaType;

[C++] public: const CompareOptions IgnoreKanaType;

[VB] Public Const IgnoreKanaType As CompareOptions

[JScript] public var IgnoreKanaType : CompareOptions;

Description

Indicates that the string comparison must ignore the Kana type. Kana type refers to Japanese hiragana and katakana characters, which represent phonetic sounds in the Japanese language. Hiragana is used for native Japanese expressions and words, while katakana is used for words borrowed from other languages, such as "computer" or "internet". A phonetic sound can be expressed in both hiragana and katakana. If this value is selected, the hiragana character for one sound is considered equal to the katakana character for the same sound.

ToString

[C#] public const CompareOptions IgnoreNonSpace;

[C++] public: const CompareOptions IgnoreNonSpace;

[VB] Public Const IgnoreNonSpace As CompareOptions

[JScript] public var IgnoreNonSpace : CompareOptions;

Description

Indicates that the string comparison must ignore nonspacing combining characters, such as diacritics. The Unicode Standard defines combining characters as characters that are combined with base characters to produce a new character.

Non-spacing combining characters do not occupy a spacing position by themselves when rendered. For more information on non-spacing combining characters, see The Unicode Standard at <http://www.unicode.org>.

ToString

[C#] public const CompareOptions IgnoreSymbols;

[C++] public: const CompareOptions IgnoreSymbols;

1 [VB] Public Const IgnoreSymbols As CompareOptions

2 [JScript] public var IgnoreSymbols : CompareOptions;

3
4 *Description*

5 Indicates that the string comparison must ignore symbols, such as white-
6 space characters, punctuation, currency symbols, the percent sign, mathematical
7 symbols, the ampersand, and so on.

8 ToString

9
10 [C#] public const CompareOptions IgnoreWidth;

11 [C++] public: const CompareOptions IgnoreWidth;

12 [VB] Public Const IgnoreWidth As CompareOptions

13 [JScript] public var IgnoreWidth : CompareOptions;

14
15 *Description*

16 Indicates that the string comparison must ignore the character width. For
17 example, Japanese katakana characters can be written as full-width or half-width
18 and, if this value is selected, the katakana characters written as full-width are
19 considered equal to the same characters written in half-width.

20 ToString

21
22 [C#] public const CompareOptions None;

23 [C++] public: const CompareOptions None;

24 [VB] Public Const None As CompareOptions

25 [JScript] public var None : CompareOptions;

1
2 *Description*

3 Indicates the default option settings for string comparisons.

4 ToString

5
6 [C#] public const CompareOptions Ordinal;

7 [C++] public: const CompareOptions Ordinal;

8 [VB] Public Const Ordinal As CompareOptions

9 [JScript] public var Ordinal : CompareOptions;

10
11 *Description*

12 Indicates that the string comparison must be done using the Unicode values
13 of each character, which is a fast comparison but is culture-insensitive. A string
14 starting with "U+xxxx" comes before a string starting with "U+yyyy", if xxxx is
15 less than yyyy.

16 ToString

17
18 [C#] public const CompareOptions StringSort;

19 [C++] public: const CompareOptions StringSort;

20 [VB] Public Const StringSort As CompareOptions

21 [JScript] public var StringSort : CompareOptions;

22
23 *Description*

Indicates that the string comparison must use the string sort method, where the hyphen and the apostrophe, as well as other non-alphanumeric characters, come before alphanumeric symbols.

CultureInfo class (System.Globalization)

ToString

Description

Represents information about a specific culture including the names of the culture, the writing system, and the calendar used, as well as methods for common operations, such as formatting dates and sorting strings.

The **System.Globalization.CultureInfo** class holds culture-specific information, such as the associated language, sublanguage, country/region, calendar, and cultural conventions. This class also provides the information required for culture-specific operations, such as casing, formatting dates and numbers, and comparing strings.

CultureInfo

Example Syntax:

ToString

[C#] public CultureInfo(int culture);

[C++] public: CultureInfo(int culture);

[VB] Public Sub New(ByVal culture As Integer)

[JScript] public function CultureInfo(culture : int);

Description

Initializes a new instance of the **System.Globalization.CultureInfo** class based on the culture specified by the culture identifier.

The *culture* parameter is mapped to the corresponding National Language Support (NLS) locale identifier. The value of the *culture* parameter becomes the value of the **System.Globalization.CultureInfo.LCID** property of the new instance. A predefined **System.Globalization.CultureInfo** identifier or the **System.Globalization.CultureInfo.LCID** of an existing **System.Globalization.CultureInfo** instance.

CultureInfo

Example Syntax:

ToString

[C#] public CultureInfo(string name);

[C++] public: CultureInfo(String* name);

[VB] Public Sub New(ByVal name As String)

[JScript] public function CultureInfo(name : String); Initializes a new instance of the **System.Globalization.CultureInfo** class.

Description

Initializes a new instance of the **System.Globalization.CultureInfo** class based on the culture specified by name.

The **System.Globalization.CultureInfo** names follow the RFC 1766 standard in the format "-", where is a lowercase two-letter code derived from ISO

639-1 and is an uppercase two-letter code derived from ISO 3166. For example, U.S. English is "en-US". The predefined **System.Globalization.CultureInfo** names are listed in the **System.Globalization.CultureInfo** class topic. A predefined **System.Globalization.CultureInfo** name or the name of an existing **System.Globalization.CultureInfo** instance.

CultureInfo

Example Syntax:

ToString

[C#] public CultureInfo(int culture, bool useUserOverride);

[C++] public: CultureInfo(int culture, bool useUserOverride);

[VB] Public Sub New(ByVal culture As Integer, ByVal useUserOverride As Boolean)

[JScript] public function CultureInfo(culture : int, useUserOverride : Boolean);

Description

Initializes a new instance of the **System.Globalization.CultureInfo** class based on the culture specified by the culture identifier and on the Boolean that specifies whether to use the user-selected culture settings from the system.

The *culture* parameter is mapped to the corresponding National Language Support (NLS) locale identifier. The value of the *culture* parameter becomes the value of the **System.Globalization.CultureInfo.LCID** property of the new instance. A predefined **System.Globalization.CultureInfo** identifier or the **System.Globalization.CultureInfo.LCID** of an existing

System.Globalization.CultureInfo instance. A Boolean that denotes whether to use the user-selected culture settings (**true**) or the default culture settings (**false**).

CultureInfo

Example Syntax:

ToString

[C#] public CultureInfo(string name, bool useUserOverride);

[C++] public: CultureInfo(String* name, bool useUserOverride);

[VB] Public Sub New(ByVal name As String, ByVal useUserOverride As Boolean)

[JScript] public function CultureInfo(name : String, useUserOverride : Boolean);

Description

Initializes a new instance of the **System.Globalization.CultureInfo** class based on the culture specified by name and on the Boolean that specifies whether to use the user-selected culture settings from the system.

The **System.Globalization.CultureInfo** names follow the RFC 1766 standard in the format "-", where is a lowercase two-letter code derived from ISO 639-1 and is an uppercase two-letter code derived from ISO 3166. For example, U.S. English is "en-US". The predefined **System.Globalization.CultureInfo** names are listed in the **System.Globalization.CultureInfo** class topic. A predefined **System.Globalization.CultureInfo** name or the name of an existing **System.Globalization.CultureInfo** instance. A Boolean that denotes whether to use the user-selected culture settings (**true**) or the default culture settings (**false**).

Calendar

ToString

[C#] public virtual Calendar Calendar {get;}

[C++] public: __property virtual Calendar* get_Calendar();

[VB] Overridable Public ReadOnly Property Calendar As Calendar

[JScript] public function get Calendar() : Calendar;

Description

Gets the default calendar used by the culture.

The **System.Globalization.CultureInfo.DateTimeFormat** property is an instance of the **System.Globalization.DateTimeFormatInfo** class which includes properties that allow users to customize the date and time formatting associated with a specific **System.Globalization.Calendar**.

CompareInfo

ToString

[C#] public virtual CompareInfo CompareInfo {get;}

[C++] public: __property virtual CompareInfo* get_CompareInfo();

[VB] Overridable Public ReadOnly Property CompareInfo As CompareInfo

[JScript] public function get CompareInfo() : CompareInfo;

Description

Gets the **System.Globalization.CompareInfo** instance that defines how to compare strings for the culture.

CurrentCulture

ToString

[C#] public static CultureInfo CurrentCulture {get;}

[C++] public: __property static CultureInfo* get_CurrentCulture();

[VB] Public Shared ReadOnly Property CurrentCulture As CultureInfo

[JScript] public static function get CurrentCulture() : CultureInfo;

Description

Gets the **System.Globalization.CultureInfo** instance that represents the culture used by the current thread.

The culture is a property of the executing thread. This read-only property returns **System.Threading.Thread.CurrentCulture** . When a thread is started, its culture is initially determined by using GetUserDefaultLCID from the Windows API. To change the culture used by a thread, set **System.Threading.Thread.CurrentCulture** to the new culture.

CurrentUICulture

ToString

[C#] public static CultureInfo CurrentUICulture {get;}

[C++] public: __property static CultureInfo* get_CurrentUICulture();

[VB] Public Shared ReadOnly Property CurrentUICulture As CultureInfo

[JScript] public static function get CurrentUICulture() : CultureInfo;

Description

Gets the **System.Globalization.CultureInfo** instance that represents the current culture used by the ResourceManager to look up culture-specific resources at run time.

The culture is a property of the executing thread. This property returns **System.Threading.Thread.CurrentUICulture** . When a thread is started, its UI culture is initially determined by using GetUserDefaultUILanguage from the Windows API. To change the UI culture used by a thread, set **System.Threading.Thread.CurrentUICulture** to the new culture.

DateTimeFormat

ToString

[C#] public virtual DateTimeFormatInfo DateTimeFormat {get; set;}

[C++] public: __property virtual DateTimeFormatInfo*

get_DateTimeFormat();public: __property virtual void

set_DateTimeFormat(DateTimeFormatInfo*);

[VB] Overridable Public Property DateTimeFormat As DateTimeFormatInfo

[JScript] public function get DateTimeFormat() : DateTimeFormatInfo;public

function set DateTimeFormat(DateTimeFormatInfo);

Description

Gets or sets a **System.Globalization.DateTimeFormatInfo** instance that defines the culturally appropriate format of displaying dates and times.

A **System.Globalization.DateTimeFormatInfo** instance can be created only for the invariant culture or for specific cultures, not for neutral cultures.

DisplayName

ToString

[C#] public virtual string DisplayName {get;}

[C++] public: __property virtual String* get_DisplayName();

[VB] Overridable Public ReadOnly Property DisplayName As String

[JScript] public function get DisplayName() : String;

Description

Gets the culture name in the format "()" in the localized language of the .NET Framework.

For example, if the .NET Framework English version is installed, the **System.Globalization.CultureInfo.DisplayName** for the specific culture U.S. English is "English (United States)". If the .NET Framework Spanish version is installed, regardless of the language that the system is set to display, the culture name is displayed in Spanish; therefore, the **System.Globalization.CultureInfo.DisplayName** for the specific culture U.S. English is "Ingles (Estados Unidos)".

EnglishName

ToString

[C#] public virtual string EnglishName {get;}

[C++] public: __property virtual String* get_EnglishName();

[VB] Overridable Public ReadOnly Property EnglishName As String

[JScript] public function get EnglishName() : String;

1
2 *Description*

3 Gets the culture name in the format "()" in English.

4 For example, the **System.Globalization.CultureInfo.EnglishName** for the
5 specific culture U.S. English is "English (United States)".

6 InstalledUICulture

7 ToString

8
9 [C#] public static CultureInfo InstalledUICulture {get;}

10 [C++] public: __property static CultureInfo* get_InstalledUICulture();

11 [VB] Public Shared ReadOnly Property InstalledUICulture As CultureInfo

12 [JScript] public static function get InstalledUICulture() : CultureInfo;

13
14 *Description*

15 Gets the **System.Globalization.CultureInfo** instance that represents the
16 culture installed with the operating system.

17 In a localized operating system, such as Japanese Windows 2000
18 Professional, this property returns the culture of the operating system. This
19 property is the equivalent of GetSystemDefaultUILanguage in the Windows API.

20 InvariantCulture

21 ToString

22
23 [C#] public static CultureInfo InvariantCulture {get;}

24 [C++] public: __property static CultureInfo* get_InvariantCulture();

25 [VB] Public Shared ReadOnly Property InvariantCulture As CultureInfo

1 [JScript] public static function get InvariantCulture() : CultureInfo;

3 *Description*

4 Gets the **System.Globalization.CultureInfo** instance that is culture-
5 independent (invariant).

6 The invariant culture is culture-insensitive. You can specify the invariant
7 culture by name using an empty string.

8 IsNeutralCulture

9 ToString

11 [C#] public virtual bool IsNeutralCulture {get;}

12 [C++] public: __property virtual bool get_IsNeutralCulture();

13 [VB] Overridable Public ReadOnly Property IsNeutralCulture As Boolean

14 [JScript] public function get IsNeutralCulture() : Boolean;

16 *Description*

17 Determines whether the current **System.Globalization.CultureInfo**
18 instance is a neutral culture.

19 A neutral culture is a culture that is associated with a language but not with
20 a country/region. A specific culture is a culture that is associated with a language
21 and a country/region. For example, "fr" is a neutral culture and "fr-FR" is a
22 specific culture. Note that "zh-CHS" (Simplified Chinese) and "zh-CHT"
23 (Traditional Chinese) are neutral cultures.

24 IsReadOnly

25 ToString

1
2 [C#] public bool IsReadOnly {get;}

3 [C++] public: __property bool get_IsReadOnly();

4 [VB] Public ReadOnly Property IsReadOnly As Boolean

5 [JScript] public function get IsReadOnly() : Boolean;

6
7 *Description*

8 Gets a value indicating whether the current

9 **System.Globalization.CultureInfo** instance is read-only.

10 LCID

11 ToString

12
13 [C#] public virtual int LCID {get;}

14 [C++] public: __property virtual int get_LCID();

15 [VB] Overridable Public ReadOnly Property LCID As Integer

16 [JScript] public function get LCID() : int;

17
18 *Description*

19 Gets the culture identifier for the current

20 **System.Globalization.CultureInfo** instance.

21 The culture identifier is mapped to the corresponding National Language

22 Support (NLS) locale identifier.

23 Name

24 ToString

1
2 [C#] public virtual string Name {get;}

3 [C++] public: __property virtual String* get_Name();

4 [VB] Overridable Public ReadOnly Property Name As String

5 [JScript] public function get Name() : String;

6
7 *Description*

8 Gets the culture name in the format "-".

9 The **System.Globalization.CultureInfo** names follow the RFC 1766
10 standard in the format "-", where is a lowercase two-letter code derived from ISO
11 639-1 and is an uppercase two-letter code derived from ISO 3166. For example,
12 the **System.Globalization.CultureInfo.Name** for the specific culture U.S.

13 English is "en-US".

14 NativeName

15 ToString

16
17 [C#] public virtual string NativeName {get;}

18 [C++] public: __property virtual String* get_NativeName();

19 [VB] Overridable Public ReadOnly Property NativeName As String

20 [JScript] public function get NativeName() : String;

21
22 *Description*

23 Gets the culture name in the format "()" in the language that the culture is
24 set to display.

The culture's full name might not display properly if the system is not set to display the culture's language correctly. For example, if the **System.Globalization.CultureInfo.Name** is "ja-JP" for Japanese (Japan), **System.Globalization.CultureInfo.NativeName** will not display correctly on a system that is set to English only. However, multilingual operating systems, such as Windows 2000, display **System.Globalization.CultureInfo.NativeName** correctly.

NumberFormat

ToString

```
[C#] public virtual NumberFormatInfo NumberFormat {get; set;}
```

```
[C++] public: __property virtual NumberFormatInfo*
```

```
get_NumberFormat();public: __property virtual void
```

```
set_NumberFormat(NumberFormatInfo*);
```

```
[VB] Overridable Public Property NumberFormat As NumberFormatInfo
```

```
[JScript] public function get NumberFormat() : NumberFormatInfo;public
```

```
function set NumberFormat(NumberFormatInfo);
```

Description

Gets or sets a **System.Globalization.NumberFormatInfo** instance that defines the culturally appropriate format of displaying numbers, currency, and percentage.

A **System.Globalization.NumberFormatInfo** instance can be created only for the invariant culture or for specific cultures, not for neutral cultures.

OptionalCalendars

ToString

[C#] public virtual Calendar[] OptionalCalendars {get;}
[C++] public: __property virtual Calendar* get_OptionalCalendars();
[VB] Overridable Public ReadOnly Property OptionalCalendars As Calendar ()
[JScript] public function get OptionalCalendars() : Calendar[];

Description

Gets the list of optional calendars that can be used by the culture.

Optional calendars are other calendars that can be used with the culture represented by the current **System.Globalization.CultureInfo** instance.

Parent

ToString

[C#] public virtual CultureInfo Parent {get;}
[C++] public: __property virtual CultureInfo* get_Parent();
[VB] Overridable Public ReadOnly Property Parent As CultureInfo
[JScript] public function get Parent() : CultureInfo;

Description

Gets the **System.Globalization.CultureInfo** instance that represents the parent culture of the current **System.Globalization.CultureInfo** instance.

A parent culture is a higher level culture that encompasses a more limited set of information that is common among its children. For example, the parent culture of "en-US" is "en"; the parent culture of "en" is the invariant culture.

1 TextInfo

2 ToString

3
4 [C#] public virtual TextInfo TextInfo {get;}

5 [C++] public: __property virtual TextInfo* get_TextInfo();

6 [VB] Overridable Public ReadOnly Property TextInfo As TextInfo

7 [JScript] public function get TextInfo() : TextInfo;

8
9 *Description*

10 Gets the **System.Globalization.TextInfo** instance that defines the writing
11 system associated with the culture.

12 The **System.Globalization.CultureInfo.TextInfo** property provides
13 culture-specific casing information for strings.

14 ThreeLetterISOLanguageName

15 ToString

16
17 [C#] public virtual string ThreeLetterISOLanguageName {get;}

18 [C++] public: __property virtual String* get_ThreeLetterISOLanguageName();

19 [VB] Overridable Public ReadOnly Property ThreeLetterISOLanguageName As
20 String

21 [JScript] public function get ThreeLetterISOLanguageName() : String;

22
23 *Description*

24 Gets the ISO 639-2 three-letter code for the language of the current
25 **System.Globalization.CultureInfo** instance.

For example, the three-letter abbreviation for English is "eng".

ThreeLetterWindowsLanguageName

ToString

```
[C#] public virtual string ThreeLetterWindowsLanguageName {get;}
```

```
[C++] public: __property virtual String*
```

```
get_ThreeLetterWindowsLanguageName();
```

```
[VB] Overridable Public ReadOnly Property ThreeLetterWindowsLanguageName
```

```
As String
```

```
[JScript] public function get ThreeLetterWindowsLanguageName() : String;
```

Description

Gets the three-letter code for the language as defined in the Windows API.

For example, the three-letter code for English (U.S.) as defined in the Windows API is "enu".

TwoLetterISOLanguageName

ToString

```
[C#] public virtual string TwoLetterISOLanguageName {get;}
```

```
[C++] public: __property virtual String* get_TwoLetterISOLanguageName();
```

```
[VB] Overridable Public ReadOnly Property TwoLetterISOLanguageName As
```

```
String
```

```
[JScript] public function get TwoLetterISOLanguageName() : String;
```

Description

1 Gets the ISO 639-1 two-letter code for the language of the current
2 **System.Globalization.CultureInfo** instance.

3 For example, the two-letter abbreviation for English is "en".

4 UseUserOverride

5 ToString

6
7 [C#] public bool UseUserOverride {get;}

8 [C++] public: __property bool get _UseUserOverride();

9 [VB] Public ReadOnly Property UseUserOverride As Boolean

10 [JScript] public function get UseUserOverride() : Boolean;

11
12 *Description*

13 Gets a value indicating whether the current
14 **System.Globalization.CultureInfo** instance uses the user-selected culture
15 settings.

16 The user might choose to override some of the values associated with the
17 default culture of the system through the Regional and Language Options (or
18 Regional Options or Regional Settings) applet in Windows Control Panel. For
19 example, the user might choose to display the date in a different format. This
20 property denotes whether the current **System.Globalization.CultureInfo** instance
21 uses those overrides (**true**) or whether it uses the default values (**false**) of the
22 culture settings.

23 ClearCachedData

24
25 [C#] public void ClearCachedData();

1 [C++] public: void ClearCachedData();

2 [VB] Public Sub ClearCachedData()

3 [JScript] public function ClearCachedData();

4
5 *Description*

6 Refreshes cached culture-related information.

7 Information, such as the default culture and format patterns, is cached the
8 first time it is requested. However, that information can change during the life of
9 the **System.AppDomain** , for example, when the user modifies the Regional and
10 Language Options (or Regional Options or Regional Settings) applet in Windows
11 Control Panel. The **System.Globalization.CultureInfo** class does not detect
12 changes in the system settings automatically. Use the
13 **System.Globalization.CultureInfo.ClearCachedData** method to refresh that
14 information in the **System.Globalization.CultureInfo** class, based on the current
15 system settings.

16 Clone

17
18 [C#] public virtual object Clone();

19 [C++] public: virtual Object* Clone();

20 [VB] Overridable Public Function Clone() As Object

21 [JScript] public function Clone() : Object;

22
23 *Description*

24 Creates a copy of the current **System.Globalization.CultureInfo** instance.

25 *Return Value:* A copy of the current **System.Globalization.CultureInfo** instance.

The clone is writable even if the original instance is read-only; therefore, the properties of the clone can be modified.

CreateSpecificCulture

[C#] public static CultureInfo CreateSpecificCulture(string name);

[C++] public: static CultureInfo* CreateSpecificCulture(String* name);

[VB] Public Shared Function CreateSpecificCulture(ByVal name As String) As CultureInfo

[JScript] public static function CreateSpecificCulture(name : String) : CultureInfo;

Description

Creates a **System.Globalization.CultureInfo** instance that represents the specific culture that is associated with the specified name.

Return Value: A **System.Globalization.CultureInfo** instance that represents the invariant culture, if *name* is "" (invariant culture).

The invariant culture is culture-insensitive. You can specify the invariant culture by name using an empty string. A predefined **System.Globalization.CultureInfo** name or the name of an existing **System.Globalization.CultureInfo** instance.

Equals

[C#] public override bool Equals(object value);

[C++] public: bool Equals(Object* value);

[VB] Overrides Public Function Equals(ByVal value As Object) As Boolean

[JScript] public override function Equals(value : Object) : Boolean;

Description

Determines whether the specified **System.Object** is the same culture as the current **System.Globalization.CultureInfo** instance.

Return Value: **true** if the specified **System.Object** is the same culture as the current **System.Globalization.CultureInfo** instance; otherwise, **false** .

This method overrides **System.Object.Equals(System.Object)** . The **System.Object** to compare with the current **System.Globalization.CultureInfo** instance.

GetCultures

[C#] public static CultureInfo[] GetCultures(CultureTypes types);

[C++] public: static CultureInfo* GetCultures(CultureTypes types) [];

[VB] Public Shared Function GetCultures(ByVal types As CultureTypes) As CultureInfo()

[JScript] public static function GetCultures(types : CultureTypes) : CultureInfo[];

Description

Gets the list of supported cultures filtered by the specified **System.Globalization.CultureTypes** .

Return Value: An array of **System.Globalization.CultureInfo** objects that represent the supported cultures filtered by the specified **System.Globalization.CultureTypes** . A combination of **System.Globalization.CultureTypes** values that filter the cultures to retrieve.

GetFormat

1
2 [C#] public virtual object GetFormat(Type formatType);

3 [C++] public: virtual Object* GetFormat(Type* formatType);

4 [VB] Overridable Public Function GetFormat(ByVal formatType As Type) As
5 Object

6 [JScript] public function GetFormat(formatType : Type) : Object;

7
8 *Description*

9 Gets an object that defines how to format the specified type.

10 *Return Value:* A **System.Globalization.NumberFormatInfo** object containing
11 the default number format information for the current

12 **System.Globalization.CultureInfo** instance, if *formatType* is the **System.Type**
13 object for the **System.Globalization.NumberFormatInfo** class.

14 **System.Globalization.CultureInfo.GetFormat(System.Type)**

15 implements **System.IFormatProvider.GetFormat(System.Type)** . The

16 **System.Type** for which to get a formatting object. This method only supports the

17 **System.Globalization.NumberFormatInfo** and

18 **System.Globalization.DateTimeFormatInfo** types.

19 GetHashCode

20
21 [C#] public override int GetHashCode();

22 [C++] public: int GetHashCode();

23 [VB] Overrides Public Function GetHashCode() As Integer

24 [JScript] public override function GetHashCode() : int;

1
2 *Description*

3 Serves as a hash function for the current

4 **System.Globalization.CultureInfo** instance, suitable for use in hashing
5 algorithms and data structures, such as a hash table.

6 *Return Value:* A hash code for the current **System.Globalization.CultureInfo**
7 instance.

8 This method overrides **System.Object.GetHashCode** .

9 **ReadOnly**

10
11 [C#] public static CultureInfo ReadOnly(CultureInfo ci);

12 [C++] public: static CultureInfo* ReadOnly(CultureInfo* ci);

13 [VB] Public Shared Function ReadOnly(ByVal ci As CultureInfo) As CultureInfo

14 [JScript] public static function ReadOnly(ci : CultureInfo) : CultureInfo;

15
16 *Description*

17 Returns a read-only wrapper around the specified

18 **System.Globalization.CultureInfo** instance.

19 *Return Value:* A read-only **System.Globalization.CultureInfo** wrapper around *ci*

20 .

21 This wrapper prevents any modifications to *ci* , to the *ci* . The

22 **System.Globalization.CultureInfo** instance to wrap.

23 **ToString**

24
25 [C#] public override string ToString();

1 [C++] public: String* ToString();

2 [VB] Overrides Public Function ToString() As String

3 [JScript] public override function ToString() : String;

4
5 *Description*

6 Returns a **System.String** containing the name of the current
7 **System.Globalization.CultureInfo** instance in the format "-".

8 *Return Value:* A **System.String** containing the name of the current
9 **System.Globalization.CultureInfo** instance in the format "-", where is a
10 lowercase two-letter code derived from ISO 639-1 and is an uppercase two-letter
11 code derived from ISO 3166.

12 This method overrides **System.Object.ToString** .

13 CultureTypes enumeration (System.Globalization)

14 ToString

15
16
17 *Description*

18 Defines the types of culture lists that can be retrieved using
19 **System.Globalization.CultureInfo.GetCultures(System.Globalization.Culture**
20 **Types)** .

21 These culture types serve as a filter that limits which cultures are returned
22 by

23 **System.Globalization.CultureInfo.GetCultures(System.Globalization.Culture**
24 **Types)** .

25 ToString

1
2 [C#] public const CultureTypes AllCultures;

3 [C++] public: const CultureTypes AllCultures;

4 [VB] Public Const AllCultures As CultureTypes

5 [JScript] public var AllCultures : CultureTypes;

6
7 *Description*

8 Refers to all cultures.

9 ToString

10
11 [C#] public const CultureTypes InstalledWin32Cultures;

12 [C++] public: const CultureTypes InstalledWin32Cultures;

13 [VB] Public Const InstalledWin32Cultures As CultureTypes

14 [JScript] public var InstalledWin32Cultures : CultureTypes;

15
16 *Description*

17 Refers to all cultures that are installed in the system.

18 ToString

19
20 [C#] public const CultureTypes NeutralCultures;

21 [C++] public: const CultureTypes NeutralCultures;

22 [VB] Public Const NeutralCultures As CultureTypes

23 [JScript] public var NeutralCultures : CultureTypes;

24
25 *Description*

Refers to cultures that are associated with a language but are not specific to a country/region. The names of these cultures consist of the lowercase two-letter code derived from ISO 639-1. For example: "en".

ToString

[C#] public const CultureTypes SpecificCultures;

[C++] public: const CultureTypes SpecificCultures;

[VB] Public Const SpecificCultures As CultureTypes

[JScript] public var SpecificCultures : CultureTypes;

Description

Refers to cultures that are specific to a country/region. The names of these cultures follow the RFC 1766 standard in the format "-", where is a lowercase two-letter code derived from ISO 639-1 and is an uppercase two-letter code derived from ISO 3166. For example, "en-US".

DateTimeFormatInfo class (System.Globalization)

ToString

Description

Defines how **System.DateTime** values are formatted and displayed, depending on the culture.

This class contains information, such as date patterns, time patterns, and AM/PM designators.

DateTimeFormatInfo

Example Syntax:

ToString

[C#] public DateTimeFormatInfo();

[C++] public: DateTimeFormatInfo();

[VB] Public Sub New()

[JScript] public function DateTimeFormatInfo();

Description

Initializes a new writable instance of the **System.Globalization.DateTimeFormatInfo** class that is culture-independent (invariant).

The properties of this instance can be modified with user-defined patterns.

AbbreviatedDayNames

ToString

[C#] public string[] AbbreviatedDayNames {get; set;}

[C++] public: __property String* get_AbbreviatedDayNames();public: __property
void set_AbbreviatedDayNames(String* __gc[]);

[VB] Public Property AbbreviatedDayNames As String ()

[JScript] public function get AbbreviatedDayNames() : String[];public function set
AbbreviatedDayNames(String[]);

Description

1 Gets or sets a one-dimensional array of type **System.String** containing the
2 abbreviated names of the days of the week.

3 If setting this property, the array must be one-dimensional and must have
4 exactly seven elements.

5 AbbreviatedMonthNames

6 ToString

7
8 [C#] public string[] AbbreviatedMonthNames {get; set;}

9 [C++] public: __property String* get_AbbreviatedMonthNames();public:

10 __property void set_AbbreviatedMonthNames(String* __gc[]);

11 [VB] Public Property AbbreviatedMonthNames As String ()

12 [JScript] public function get AbbreviatedMonthNames() : String[];public function

13 set AbbreviatedMonthNames(String[]);

14
15 *Description*

16 Gets or sets a one-dimensional array of type **System.String** containing the
17 abbreviated names of the months.

18 If setting this property, the array must be one-dimensional and must have
19 exactly 13 elements.

20 AMDesignator

21 ToString

22
23 [C#] public string AMDesignator {get; set;}

24 [C++] public: __property String* get_AMDesignator();public: __property void

25 set_AMDesignator(String*);

1 [VB] Public Property AMDesignator As String

2 [JScript] public function get AMDesignator() : String;public function set

3 AMDesignator(String);

4
5 *Description*

6 Indicates the **System.String** designator for hours that are "ante meridiem"
7 (before noon).

8 If the custom pattern includes the format pattern "tt" and the time is before
9 noon, **System.DateTime.ToString** displays the value of
10 **System.Globalization.DateTimeFormatInfo.AMDesignator** in place of the "tt"
11 in the format pattern. If the custom pattern includes the format pattern "t", only the
12 first character of **System.Globalization.DateTimeFormatInfo.AMDesignator** is
13 displayed.

14 Calendar

15 ToString

16
17 [C#] public Calendar Calendar {get; set;}

18 [C++] public: __property Calendar* get_Calendar();public: __property void

19 set_Calendar(Calendar*);

20 [VB] Public Property Calendar As Calendar

21 [JScript] public function get Calendar() : Calendar;public function set

22 Calendar(Calendar);

23
24 *Description*

25 Gets or sets the calendar to use for the current culture.

The **System.Globalization.DateTimeFormatInfo.Calendar** property only accepts calendars that are valid for the current culture of the current thread. The **System.Globalization.CultureInfo.Calendar** property specifies the default calendar for the culture and the **System.Globalization.CultureInfo.OptionalCalendars** property specifies other calendars supported by the culture.

CalendarWeekRule

ToString

[C#] public CalendarWeekRule CalendarWeekRule {get; set;}

[C++] public: __property CalendarWeekRule get_CalendarWeekRule();public:

__property void set_CalendarWeekRule(CalendarWeekRule);

[VB] Public Property CalendarWeekRule As CalendarWeekRule

[JScript] public function get CalendarWeekRule() : CalendarWeekRule;public

function set CalendarWeekRule(CalendarWeekRule);

Description

Gets or sets a value that specifies which rule is used to determine the first calendar week of the year.

This property is affected if the value of the **System.Globalization.DateTimeFormatInfo.Calendar** property changes.

CurrentInfo

ToString

[C#] public static DateTimeFormatInfo CurrentInfo {get;}

1 [C++] public: __property static DateTimeFormatInfo* get_CurrentInfo();

2 [VB] Public Shared ReadOnly Property CurrentInfo As DateTimeFormatInfo

3 [JScript] public static function get CurrentInfo() : DateTimeFormatInfo;

4
5 *Description*

6 Gets a read-only **System.Globalization.DateTimeFormatInfo** instance
7 that formats values based on the current culture.

8 DateSeparator

9 ToString

10
11 [C#] public string DateSeparator {get; set;}

12 [C++] public: __property String* get_DateSeparator();public: __property void
13 set_DateSeparator(String*);

14 [VB] Public Property DateSeparator As String

15 [JScript] public function get DateSeparator() : String;public function set
16 DateSeparator(String);

17
18 *Description*

19 Indicates the **System.String** that separates the components of a date; that is,
20 the year, month, and day.

21 If the custom pattern includes the format pattern "/",

22 **System.DateTime.ToString** displays the value of

23 **System.Globalization.DateTimeFormatInfo.DateSeparator** in place of the "/"
24 in the format pattern.

25 DayNames

ToString

[C#] public string[] DayNames {get; set;}

[C++] public: __property String* get_DayNames();public: __property void

set_DayNames(String* __gc[]);

[VB] Public Property DayNames As String ()

[JScript] public function get DayNames() : String[];public function set

DayNames(String[]);

Description

Gets or sets a one-dimensional array of type **System.String** containing the full names of the days of the week.

If setting this property, the array must be one-dimensional and must have exactly seven elements.

FirstDayOfWeek

ToString

[C#] public DayOfWeek FirstDayOfWeek {get; set;}

[C++] public: __property DayOfWeek get_FirstDayOfWeek();public: __property

void set_FirstDayOfWeek(DayOfWeek);

[VB] Public Property FirstDayOfWeek As DayOfWeek

[JScript] public function get FirstDayOfWeek() : DayOfWeek;public function set

FirstDayOfWeek(DayOfWeek);

Description

Indicates the first day of the week.

This property is affected if the value of the

System.Globalization.DateTimeFormatInfo.Calendar property changes.

FullDateTimePattern

ToString

[C#] public string FullDateTimePattern {get; set;}

[C++] public: __property String* get_FullDateTimePattern();public: __property
void set_FullDateTimePattern(String*);

[VB] Public Property FullDateTimePattern As String

[JScript] public function get FullDateTimePattern() : String;public function set
FullDateTimePattern(String);

Description

Indicates the format pattern for a long date and long time value, which is associated with the 'F' format character.

See **System.Globalization.DateTimeFormatInfo** for patterns that can be combined to construct custom patterns; for example, "dddd, dd MMMM yyyy HH:mm:ss".

InvariantInfo

ToString

[C#] public static DateTimeFormatInfo InvariantInfo {get;}

[C++] public: __property static DateTimeFormatInfo* get_InvariantInfo();

[VB] Public Shared ReadOnly Property InvariantInfo As DateTimeFormatInfo

1 [JScript] public static function get InvariantInfo() : DateTimeFormatInfo;

3 *Description*

4 Gets the default read-only **System.Globalization.DateTimeFormatInfo**
5 instance that is culture-independent (invariant).

6 This property does not change regardless of the current culture.

7 IsReadOnly

8 ToString

10 [C#] public bool IsReadOnly {get;}

11 [C++] public: __property bool get_IsReadOnly();

12 [VB] Public ReadOnly Property IsReadOnly As Boolean

13 [JScript] public function get IsReadOnly() : Boolean;

15 *Description*

16 Gets a value indicating whether the
17 **System.Globalization.DateTimeFormatInfo** is read-only.

18 LongDatePattern

19 ToString

21 [C#] public string LongDatePattern {get; set;}

22 [C++] public: __property String* get_LongDatePattern();public: __property void
23 set_LongDatePattern(String*);

24 [VB] Public Property LongDatePattern As String

25 [JScript] public function get LongDatePattern() : String;public function set

1 LongDatePattern(String);

2
3 *Description*

4 Indicates the format pattern for a long date value, which is associated with
5 the 'D' format character.

6 See **System.Globalization.DateTimeFormatInfo** for patterns that can be
7 combined to construct custom patterns; for example, "dddd, dd MMMM yyyy".

8 LongTimePattern

9 ToString

10
11 [C#] public string LongTimePattern {get; set;}

12 [C++] public: __property String* get_LongTimePattern();public: __property void
13 set_LongTimePattern(String*);

14 [VB] Public Property LongTimePattern As String

15 [JScript] public function get LongTimePattern() : String;public function set

16 LongTimePattern(String);

17
18 *Description*

19 Indicates the format pattern for a long time value, which is associated with
20 the 'T' format character.

21 See **System.Globalization.DateTimeFormatInfo** for patterns that can be
22 combined to construct custom patterns; for example, "HH:mm:ss".

23 MonthDayPattern

24 ToString

1
2 [C#] public string MonthDayPattern {get; set;}

3 [C++] public: __property String* get_MonthDayPattern();public: __property void
4 set_MonthDayPattern(String*);

5 [VB] Public Property MonthDayPattern As String

6 [JScript] public function get MonthDayPattern() : String;public function set
7 MonthDayPattern(String);

8
9 *Description*

10 Indicates the format pattern for a month and day value, which is associated
11 with the 'm' and 'M' format characters.

12 See **System.Globalization.DateTimeFormatInfo** for patterns that can be
13 combined to construct custom patterns; for example, "MMMM dd".

14 MonthNames

15 ToString

16
17 [C#] public string[] MonthNames {get; set;}

18 [C++] public: __property String* get_MonthNames();public: __property void
19 set_MonthNames(String* __gc[]);

20 [VB] Public Property MonthNames As String ()

21 [JScript] public function get MonthNames() : String[];public function set
22 MonthNames(String[]);

23
24 *Description*

Gets or sets a one-dimensional array of type **System.String** containing the full names of the months.

If setting this property, the array must be one-dimensional and must have exactly 13 elements.

PMDesignator

ToString

[C#] public string PMDesignator {get; set;}

[C++] public: __property String* get_PMDesignator();public: __property void set_PMDesignator(String*);

[VB] Public Property PMDesignator As String

[JScript] public function get PMDesignator() : String;public function set PMDesignator(String);

Description

Indicates the **System.String** designator for hours that are "post meridiem" (after noon).

If the custom pattern includes the format pattern "tt" and the time is after noon, **System.DateTime.ToString** displays the value of **System.Globalization.DateTimeFormatInfo.PMDesignator** in place of the "tt" in the format pattern. If the custom pattern includes the format pattern "t", only the first character of **System.Globalization.DateTimeFormatInfo.PMDesignator** is displayed.

RFC1123Pattern

ToString

1
2 [C#] public string RFC1123Pattern {get;}

3 [C++] public: __property String* get_RFC1123Pattern();

4 [VB] Public ReadOnly Property RFC1123Pattern As String

5 [JScript] public function get RFC1123Pattern() : String;

6
7 *Description*

8 Gets the format pattern for a time value, which is based on the Internet
9 Engineering Task Force (IETF) Request for Comments (RFC) 1123 specification
10 and is associated with the 'r' and 'R' format characters.

11 ShortDatePattern

12 ToString

13
14 [C#] public string ShortDatePattern {get; set;}

15 [C++] public: __property String* get_ShortDatePattern();public: __property void
16 set_ShortDatePattern(String*);

17 [VB] Public Property ShortDatePattern As String

18 [JScript] public function get ShortDatePattern() : String;public function set
19 ShortDatePattern(String);

20
21 *Description*

22 Indicates the format pattern for a short date value, which is associated with
23 the 'd' format character.

24 See **System.Globalization.DateTimeFormatInfo** for patterns that can be
25 combined to construct custom patterns; for example, "MM/dd/yyyy".

ShortTimePattern

ToString

[C#] public string ShortTimePattern {get; set;}

[C++] public: __property String* get_ShortTimePattern();public: __property void
set_ShortTimePattern(String*);

[VB] Public Property ShortTimePattern As String

[JScript] public function get ShortTimePattern() : String;public function set
ShortTimePattern(String);

Description

Indicates the format pattern for a short time value, which is associated with
the 't' format character.

See **System.Globalization.DateTimeFormatInfo** for patterns that can be
combined to construct custom patterns; for example, "HH:mm".

SortableDateTimePattern

ToString

[C#] public string SortableDateTimePattern {get;}

[C++] public: __property String* get_SortableDateTimePattern();

[VB] Public ReadOnly Property SortableDateTimePattern As String

[JScript] public function get SortableDateTimePattern() : String;

Description

Gets the format pattern for a sortable date and time value, which is associated with the 's' format character.

TimeSeparator

ToString

[C#] public string TimeSeparator {get; set;}

[C++] public: __property String* get_TimeSeparator();public: __property void set_TimeSeparator(String*);

[VB] Public Property TimeSeparator As String

[JScript] public function get TimeSeparator() : String;public function set TimeSeparator(String);

Description

Indicates the **System.String** that separates the components of time; that is, the hour, minutes, and seconds.

If the custom pattern includes the format pattern ":", **System.DateTime.ToString** displays the value of **System.Globalization.DateTimeFormatInfo.TimeSeparator** in place of the ":" in the format pattern.

UniversalSortableDateTimePattern

ToString

[C#] public string UniversalSortableDateTimePattern {get;}

[C++] public: __property String* get_UniversalSortableDateTimePattern();

[VB] Public ReadOnly Property UniversalSortableDateTimePattern As String

1 [JScript] public function get UniversalSortableDateTimePattern() : String;

3 *Description*

4 Gets the format pattern for a universal sortable date and time value, which
5 is associated with the 'u' and 'U' format characters.

6 YearMonthPattern

7 ToString

9 [C#] public string YearMonthPattern {get; set;}

10 [C++] public: __property String* get_YearMonthPattern();public: __property void
11 set_YearMonthPattern(String*);

12 [VB] Public Property YearMonthPattern As String

13 [JScript] public function get YearMonthPattern() : String;public function set
14 YearMonthPattern(String);

16 *Description*

17 Indicates the format pattern for a year and month value, which is associated
18 with the 'y' and 'Y' format characters.

19 See **System.Globalization.DateTimeFormatInfo** for patterns that can be
20 combined to construct custom patterns; for example, "yyyy MMMM".

21 Clone

23 [C#] public object Clone();

24 [C++] public: __sealed Object* Clone();

25 [VB] NotOverridable Public Function Clone() As Object

1 [JScript] public function Clone() : Object;

3 *Description*

4 Creates a shallow copy of the
5 **System.Globalization.DateTimeFormatInfo** instance.

6 *Return Value:* A new **System.Globalization.DateTimeFormatInfo** instance
7 copied from the original **System.Globalization.DateTimeFormatInfo** instance.

8 The clone is writable even if the original instance is read-only; therefore,
9 the properties of the clone can be modified with user-defined patterns.

10 GetAbbreviatedDayName

12 [C#] public string GetAbbreviatedDayName(DayOfWeek dayofweek);

13 [C++] public: String* GetAbbreviatedDayName(DayOfWeek dayofweek);

14 [VB] Public Function GetAbbreviatedDayName(ByVal dayofweek As
15 DayOfWeek) As String

16 [JScript] public function GetAbbreviatedDayName(dayofweek : DayOfWeek) :
17 String; Gets the abbreviated name of the specified day of the week.

19 *Description*

20 Gets the abbreviated name of the specified day of the week based on the
21 **System.Globalization.CultureInfo** of the current thread.

22 *Return Value:* The abbreviated name of the day of the week represented by
23 *dayofweek* .

24 For the default invariant **System.Globalization.DateTimeFormatInfo**
25 instance, this method returns a string from the

System.Globalization.GregorianCalendar : *dayofweek* Return Value Sunday
"Sun" Monday "Mon" Tuesday "Tue" Wednesday "Wed" Thursday "Thu" Friday
"Fri" Saturday "Sat" A **System.DayOfWeek** value.

GetAbbreviatedEraName

[C#] public string GetAbbreviatedEraName(int era);

[C++] public: String* GetAbbreviatedEraName(int era);

[VB] Public Function GetAbbreviatedEraName(ByVal era As Integer) As String

[JScript] public function GetAbbreviatedEraName(era : int) : String;

Description

Gets the **System.String** containing the abbreviated name of the specified era, if an abbreviation exists.

Return Value: A **System.String** containing the abbreviated name of the specified era, if an abbreviation exists.

The valid values for *era* are listed in the **System.Globalization.Calendar.Eras** property of the appropriate class derived from **System.Globalization.Calendar** . For example:

System.Globalization.JapaneseCalendar.Eras displays a list of eras that are supported by this implementation. The integer representing the era.

GetAbbreviatedMonthName

[C#] public string GetAbbreviatedMonthName(int month);

[C++] public: String* GetAbbreviatedMonthName(int month);

[VB] Public Function GetAbbreviatedMonthName(ByVal month As Integer) As

String

[JScript] public function GetAbbreviatedMonthName(month : int) : String; Gets the abbreviated name of the specified month.

Description

Gets the abbreviated name of the specified month based on the **System.Globalization.CultureInfo** of the current thread.

Return Value: The abbreviated name of the month represented by *month* .

For the default invariant **System.Globalization.DateTimeFormatInfo** instance, this method returns a string from the **System.Globalization.GregorianCalendar** : *month* Return Value 1 "Jan" 2 "Feb" 3 "Mar" 4 "Apr" 5 "May" 6 "Jun" 7 "Jul" 8 "Aug" 9 "Sep" 10 "Oct" 11 "Nov" 12 "Dec" 13 "" **System.Globalization.Calendar** objects can accommodate calendars with 13 months. For 12-month calendars, the empty string is always returned as the name of the 13th month. An integer from 1 through 13 representing the name of the month to retrieve.

GetAllDateTimePatterns

[C#] public string[] GetAllDateTimePatterns();

[C++] public: String* GetAllDateTimePatterns() __gc[];

[VB] Public Function GetAllDateTimePatterns() As String()

[JScript] public function GetAllDateTimePatterns() : String[]; Gets the standard patterns in which **System.DateTime** values can be formatted.

Description

Gets all the standard patterns in which **System.DateTime** values can be formatted.

Return Value: An array containing the standard patterns in which **System.DateTime** values can be formatted.

See the summary page of the **System.Globalization.DateTimeFormatInfo** class for a list of the standard format characters and their associated patterns.

GetAllDateTimePatterns

[C#] public string[] GetAllDateTimePatterns(char format);

[C++] public: String* GetAllDateTimePatterns(__wchar_t format) __gc[];

[VB] Public Function GetAllDateTimePatterns(ByVal format As Char) As String()

[JScript] public function GetAllDateTimePatterns(format : Char) : String[];

Description

Gets all the standard patterns in which **System.DateTime** values can be formatted using the specified format character.

Return Value: An array containing the standard patterns in which **System.DateTime** values can be formatted using the specified format character.

See the summary page of the **System.Globalization.DateTimeFormatInfo** class for a list of the standard format characters and their associated patterns. A standard format character.

GetDayName

[C#] public string GetDayName(DayOfWeek dayofweek);

1 [C++] public: String* GetDayName(DayOfWeek dayofweek);

2 [VB] Public Function GetDayName(ByVal dayofweek As DayOfWeek) As String

3 [JScript] public function GetDayName(dayofweek : DayOfWeek) : String; Gets
4 the full name of the specified day of the week.

5
6 *Description*

7 Gets the full name of the specified day of the week based on the
8 **System.Globalization.CultureInfo** of the current thread.

9 *Return Value:* The full name of the day of the week represented by *dayofweek* .

10 For the default invariant **System.Globalization.DateTimeFormatInfo**
11 instance, this method returns a string from the

12 **System.Globalization.GregorianCalendar** : *dayofweek* Return Value Sunday

13 "Sunday" Monday "Monday" Tuesday "Tuesday" Wednesday "Wednesday"

14 Thursday "Thursday" Friday "Friday" Saturday "Saturday" A

15 **System.DayOfWeek** value.

16 GetEra

17
18 [C#] public int GetEra(string eraName);

19 [C++] public: int GetEra(String* eraName);

20 [VB] Public Function GetEra(ByVal eraName As String) As Integer

21 [JScript] public function GetEra(eraName : String) : int;

22
23 *Description*

24 Gets the integer representing the specified era.

25 *Return Value:* The integer representing the era, if *eraName* is valid; otherwise, -1.

The comparison with *eraName* is case-insensitive; for example, "A.D." is equivalent to "a.d.". The **System.String** containing the name of the era.

GetEraName

[C#] public string GetEraName(int era);

[C++] public: String* GetEraName(int era);

[VB] Public Function GetEraName(ByVal era As Integer) As String

[JScript] public function GetEraName(era : int) : String;

Description

Gets the **System.String** containing the name of the specified era.

Return Value: A **System.String** containing the name of the era.

The valid values for *era* are listed in the

System.Globalization.Calendar.Eras property of the appropriate class derived from **System.Globalization.Calendar** . For example:

System.Globalization.JapaneseCalendar.Eras displays a list of eras that are supported by this implementation. The integer representing the era.

GetFormat

[C#] public object GetFormat(Type formatType);

[C++] public: __sealed Object* GetFormat(Type* formatType);

[VB] NotOverridable Public Function GetFormat(ByVal formatType As Type) As Object

[JScript] public function GetFormat(formatType : Type) : Object;

1
2 *Description*

3 Gets an object of the specified type that provides a **System.DateTime**
4 formatting service.

5 *Return Value:* The current instance of the
6 **System.Globalization.DateTimeFormatInfo** class, if *formatType* is the same as
7 the type of the current instance; otherwise, **null** .

8 The Format(String, IFormatProvider) method supported by the base data
9 types invoke this method when the current instance is passed as the
10 **System.IFormatProvider** parameter. This method implements
11 **System.IFormatProvider.GetFormat(System.Type)** . The **System.Type** of the
12 required formatting service.

13 *GetInstance*

14
15 [C#] public static DateTimeFormatInfo GetInstance(IFormatProvider provider);

16 [C++] public: static DateTimeFormatInfo* GetInstance(IFormatProvider*
17 provider);

18 [VB] Public Shared Function GetInstance(ByVal provider As IFormatProvider)

19 As DateTimeFormatInfo

20 [JScript] public static function GetInstance(provider : IFormatProvider) :

21 DateTimeFormatInfo;

22
23 *Description*

24 Gets the **System.Globalization.DateTimeFormatInfo** instance associated
25 with the specified **System.IFormatProvider** .

Return Value: A **System.Globalization.DateTimeFormatInfo** instance associated with the specified **System.IFormatProvider** .

This method uses the **System.IFormatProvider.GetFormat(System.Type)** method of *formatProvider* using **System.Globalization.DateTimeFormatInfo** as the *Type* parameter. If *formatProvider* is **null** or if **System.IFormatProvider.GetFormat(System.Type)** returns **null** , this method returns **System.Globalization.DateTimeFormatInfo.CurrentInfo** . The **System.IFormatProvider** that gets the **System.Globalization.DateTimeFormatInfo** instance.

GetMonthName

[C#] public string GetMonthName(int month);

[C++] public: String* GetMonthName(int month);

[VB] Public Function GetMonthName(ByVal month As Integer) As String

[JScript] public function GetMonthName(month : int) : String; Gets the full name of the specified month.

Description

Gets the full name of the specified month based on the **System.Globalization.CultureInfo** of the current thread.

Return Value: The full name of the month represented by *month* .

For the default invariant **System.Globalization.DateTimeFormatInfo** instance, this method returns a string from the **System.Globalization.GregorianCalendar** : *month* Return Value 1 "January" 2

1 "February" 3 "March" 4 "April" 5 "May" 6 "June" 7 "July" 8 "August" 9

2 "September" 10 "October" 11 "November" 12 "December" 13 ""

3 **System.Globalization.Calendar** objects can accommodate calendars with 13
4 months. For 12-month calendars, the empty string is always returned as the name
5 of the 13th month. An integer from 1 through 13 representing the name of the
6 month to retrieve.

7 **ReadOnly**

8
9 [C#] public static DateTimeFormatInfo ReadOnly(DateTimeFormatInfo dtfi);

10 [C++] public: static DateTimeFormatInfo* ReadOnly(DateTimeFormatInfo* dtfi);

11 [VB] Public Shared Function ReadOnly(ByVal dtfi As DateTimeFormatInfo) As
12 DateTimeFormatInfo

13 [JScript] public static function ReadOnly(dtfi : DateTimeFormatInfo) :

14 DateTimeFormatInfo;

15
16 *Description*

17 Returns a read-only **System.Globalization.DateTimeFormatInfo**
18 wrapper.

19 *Return Value:* A read-only **System.Globalization.DateTimeFormatInfo** wrapper
20 around *dtfi* .

21 This wrapper prevents any modifications to *dtfi* . The
22 **System.Globalization.DateTimeFormatInfo** to wrap.

23 DateTimeStyles enumeration (System.Globalization)

24 ToString

1
2
3 *Description*

4 Defines the formatting options that customize how the
5 **System.DateTime.Parse(System.String)** and
6 **System.DateTime.ParseExact(System.String, System.String, System.IFormatP
7 rovider)** methods parse a string.

8 The **System.Globalization.DateTimeStyles.NoCurrentDateDefault** value
9 is the only value that is useful with the **System.DateTime.Parse(System.String)**
10 method, because **System.DateTime.Parse(System.String)** always ignores
11 leading, trailing, and inner white-space characters.

12 ToString

13
14 [C#] public const DateTimeStyles AdjustToUniversal;
15 [C++] public: const DateTimeStyles AdjustToUniversal;
16 [VB] Public Const AdjustToUniversal As DateTimeStyles
17 [JScript] public var AdjustToUniversal : DateTimeStyles;

18
19 *Description*

20 Indicates that the date and time must be converted to Universal Time or
21 Greenwich mean time (GMT).

22 ToString

23
24 [C#] public const DateTimeStyles AllowInnerWhite;
25 [C++] public: const DateTimeStyles AllowInnerWhite;

1 [VB] Public Const AllowInnerWhite As DateTimeStyles

2 [JScript] public var AllowInnerWhite : DateTimeStyles;

3
4 *Description*

5 Indicates that extra white-space characters in the middle of the string must
6 be ignored.

7 ToString

8
9 [C#] public const DateTimeStyles AllowLeadingWhite;

10 [C++] public: const DateTimeStyles AllowLeadingWhite;

11 [VB] Public Const AllowLeadingWhite As DateTimeStyles

12 [JScript] public var AllowLeadingWhite : DateTimeStyles;

13
14 *Description*

15 Indicates that leading white-space characters must be ignored.

16 ToString

17
18 [C#] public const DateTimeStyles AllowTrailingWhite;

19 [C++] public: const DateTimeStyles AllowTrailingWhite;

20 [VB] Public Const AllowTrailingWhite As DateTimeStyles

21 [JScript] public var AllowTrailingWhite : DateTimeStyles;

22
23 *Description*

24 Indicates that trailing white-space characters must be ignored.

25 ToString

1
2 [C#] public const DateTimeStyles AllowWhiteSpaces;
3 [C++] public: const DateTimeStyles AllowWhiteSpaces;
4 [VB] Public Const AllowWhiteSpaces As DateTimeStyles
5 [JScript] public var AllowWhiteSpaces : DateTimeStyles;
6

7 *Description*

8 Indicates that extra white-space characters anywhere in the string must be
9 ignored. This value is a combination of the
10 **System.Globalization.DateTimeStyles.AllowLeadingWhite** ,
11 **System.Globalization.DateTimeStyles.AllowTrailingWhite** , and
12 **System.Globalization.DateTimeStyles.AllowInnerWhite** values.

13 ToString
14

15 [C#] public const DateTimeStyles NoCurrentDateDefault;
16 [C++] public: const DateTimeStyles NoCurrentDateDefault;
17 [VB] Public Const NoCurrentDateDefault As DateTimeStyles
18 [JScript] public var NoCurrentDateDefault : DateTimeStyles;
19

20 *Description*

21 Indicates that, if the string does not include a date, the year, month, and day
22 will each be set to "1" instead of the current year, month, and day.

23 ToString
24

25 [C#] public const DateTimeStyles None;

1 [C++] public: const DateTimeStyles None;

2 [VB] Public Const None As DateTimeStyles

3 [JScript] public var None : DateTimeStyles;

4
5 *Description*

6 Indicates that the default formatting options must be used. This is the
7 default style for **System.DateTime.Parse(System.String)** and
8 **System.DateTime.ParseExact(System.String, System.String, System.IFormatP
9 rovider)** .

10 DaylightTime class (System.Globalization)

11 ToString

12
13
14 *Description*

15 Defines the period of daylight saving time.

16 Daylight saving time is a period during the year when the time is advanced,
17 usually by an hour, to take advantage of the extended daylight hours. At the end of
18 the period, the time is set back to the standard time.

19 DaylightTime

20 *Example Syntax:*

21 ToString

22
23 [C#] public DaylightTime(DateTime start, DateTime end, TimeSpan delta);

24 [C++] public: DaylightTime(DateTime start, DateTime end, TimeSpan delta);

25 [VB] Public Sub New(ByVal start As DateTime, ByVal end As DateTime, ByVal

1 delta As TimeSpan)

2 [JScript] public function DaylightTime(start : DateTime, end : DateTime, delta :
3 TimeSpan);

4
5 *Description*

6 Initializes a new instance of the **System.Globalization.DaylightTime**
7 class.

8 The *start* parameter becomes the value of the
9 **System.Globalization.DaylightTime.Start** property of the new instance. The *end*
10 parameter becomes the value of the **System.Globalization.DaylightTime.End**
11 property of the new instance. The *delta* parameter becomes the value of the
12 **System.Globalization.DaylightTime.Delta** property of the new instance. The
13 **System.DateTime** instance that represents the date and time when the daylight
14 saving period begins. The value must be in local time. The **System.DateTime**
15 instance that represents the date and time when the daylight saving period ends.
16 The value must be in local time. The **System.TimeSpan** instance that represents
17 the difference between the standard time and the daylight saving time in ticks.

18 Delta

19 ToString

20
21 [C#] public TimeSpan Delta {get;}

22 [C++] public: __property TimeSpan get_Delta();

23 [VB] Public ReadOnly Property Delta As TimeSpan

24 [JScript] public function get Delta() : TimeSpan;

Description

Gets the **System.TimeSpan** instance that represents the difference between the standard time and the daylight saving time.

At the start of daylight saving time, the clock time is advanced by the length of time specified in this property. At the end of daylight saving time, the clock time is set back by the length of time specified in this property.

End

ToString

[C#] public DateTime End {get;}

[C++] public: __property DateTime get_End();

[VB] Public ReadOnly Property End As DateTime

[JScript] public function get End() : DateTime;

Description

Gets the **System.DateTime** instance that represents the date and time when the daylight saving period ends.

When the daylight saving period ends, the clock time is set back to the standard time.

Start

ToString

[C#] public DateTime Start {get;}

[C++] public: __property DateTime get_Start();

1 [VB] Public ReadOnly Property Start As DateTime

2 [JScript] public function get Start() : DateTime;

3
4 *Description*

5 Gets the **System.DateTime** instance that represents the date and time when
6 the daylight saving period begins.

7 When the daylight saving period begins, the clock time is advanced by the
8 number of ticks defined in **System.Globalization.DaylightTime.Delta** to take
9 advantage of the extended daylight hours.

10 GregorianCalendar class (System.Globalization)

11 ToString

12
13
14 *Description*

15 Represents the Gregorian calendar.

16 The Gregorian calendar recognizes two eras: B.C. (before Christ) or B.C.E.
17 (before common era), and A.D. (Latin "Anno Domini", which means "in the year
18 of the Lord") or C.E. (common era). This implementation of the
19 **System.Globalization.GregorianCalendar** class recognizes only the current era
20 (A.D. or C.E.).

21 ToString

22
23 [C#] public const int AD Era;

24 [C++] public: const int AD Era;

25 [VB] Public Const AD Era As Integer

1 [JScript] public var ADera : int;

2
3 *Description*

4 Represents the current era.

5 The Gregorian calendar recognizes two eras: B.C. (before Christ) or B.C.E.
6 (before common era), and A.D. (Latin "Anno Domini", which means "in the year
7 of the Lord") or C.E. (common era). This implementation of the
8 **System.Globalization.GregorianCalendar** class recognizes only the current era
9 (A.D. or C.E.). This field always returns 1.

10 **GregorianCalendar**

11 *Example Syntax:*

12 ToString

13
14 [C#] public GregorianCalendar();

15 [C++] public: GregorianCalendar();

16 [VB] Public Sub New()

17 [JScript] public function GregorianCalendar(); Initializes a new instance of the
18 **System.Globalization.GregorianCalendar** class.

19
20 *Description*

21 Initializes a new instance of the
22 **System.Globalization.GregorianCalendar** class using the default
23 **System.Globalization.GregorianCalendarTypes** value.

24 The default **System.Globalization.GregorianCalendarTypes** value is
25 **System.Globalization.GregorianCalendarTypes.Localized** .

GregorianCalendar

Example Syntax:

ToString

[C#] public GregorianCalendar(GregorianCalendarTypes type);

[C++] public: GregorianCalendar(GregorianCalendarTypes type);

[VB] Public Sub New(ByVal type As GregorianCalendarTypes)

[JScript] public function GregorianCalendar(type : GregorianCalendarTypes);

Description

Initializes a new instance of the **System.Globalization.GregorianCalendar** class using the specified **System.Globalization.GregorianCalendarTypes** value. The **System.Globalization.GregorianCalendarTypes** value that denotes which version of the calendar to create.

CalendarType

ToString

[C#] public virtual GregorianCalendarTypes CalendarType {get; set;}

[C++] public: __property virtual GregorianCalendarTypes

get_CalendarType();public: __property virtual void

set_CalendarType(GregorianCalendarTypes);

[VB] Overridable Public Property CalendarType As GregorianCalendarTypes

[JScript] public function get CalendarType() : GregorianCalendarTypes;public

function set CalendarType(GregorianCalendarTypes);

Description

Gets or sets the **System.Globalization.GregorianCalendarTypes** value that denotes the version of the current **System.Globalization.GregorianCalendar** instance.

Eras

ToString

[C#] public override int[] Eras {get;}

[C++] public: __property virtual int get_Eras();

[VB] Overrides Public ReadOnly Property Eras As Integer ()

[JScript] public function get Eras() : int[];

Description

Gets the list of eras in the **System.Globalization.GregorianCalendar**.

The Gregorian calendar recognizes two eras: B.C. (before Christ) or B.C.E. (before common era), and A.D. (Latin "Anno Domini", which means "in the year of the Lord") or C.E. (common era). This implementation of the **System.Globalization.GregorianCalendar** class recognizes only the current era (A.D. or C.E.). This property always returns an array with only one element.

TwoDigitYearMax

ToString

[C#] public override int TwoDigitYearMax {get; set;}

[C++] public: __property virtual int get_TwoDigitYearMax();public: __property

1 virtual void set_TwoDigitYearMax(int);

2 [VB] Overrides Public Property TwoDigitYearMax As Integer

3 [JScript] public function get TwoDigitYearMax() : int;public function set

4 TwoDigitYearMax(int);

5
6 *Description*

7 Gets or sets the last year of a 100-year range that can be represented by a 2-
8 digit year.

9 This property allows a 2-digit year to be properly translated to a 4-digit
10 year. For example, if this property is set to 2029, the 100-year range is from 1930
11 to 2029; therefore, a 2-digit value of 30 is interpreted as 1930, while a 2-digit
12 value of 29 is interpreted as 2029.

13 AddMonths

14
15 [C#] public override DateTime AddMonths(DateTime time, int months);

16 [C++] public: DateTime AddMonths(DateTime time, int months);

17 [VB] Overrides Public Function AddMonths(ByVal time As DateTime, ByVal
18 months As Integer) As DateTime

19 [JScript] public override function AddMonths(time : DateTime, months : int) :

20 DateTime;

21
22 *Description*

23 Returns a **System.DateTime** that is the specified number of months away
24 from the specified **System.DateTime** .

1 *Return Value:* The **System.DateTime** that results from adding the specified
2 number of months to the specified **System.DateTime** .

3 The year part of the resulting **System.DateTime** is affected if the resulting
4 month is beyond the last month of the current year. The day part of the resulting
5 **System.DateTime** is also affected if the resulting day is not a valid day in the
6 resulting month of the resulting year; it is changed to the last valid day in the
7 resulting month of the resulting year. The time-of-day part of the resulting
8 **System.DateTime** remains the same as the specified **System.DateTime** . The
9 **System.DateTime** instance to add. The number of months to add.

10 AddWeeks

11
12 [C#] public override DateTime AddWeeks(DateTime time, int weeks);

13 [C++] public: DateTime AddWeeks(DateTime time, int weeks);

14 [VB] Overrides Public Function AddWeeks(ByVal time As DateTime, ByVal
15 weeks As Integer) As DateTime

16 [JScript] public override function AddWeeks(time : DateTime, weeks : int) :
17 DateTime;

18 19 *Description*

20 Returns a **System.DateTime** that is the specified number of weeks away
21 from the specified **System.DateTime** .

22 *Return Value:* The **System.DateTime** that results from adding the specified
23 number of weeks to the specified **System.DateTime** .
24
25

If *weeks* is negative, the resulting **System.DateTime** would be earlier than the specified **System.DateTime** . The **System.DateTime** instance to add. The number of weeks to add.

AddYears

[C#] public override DateTime AddYears(DateTime time, int years);

[C++] public: DateTime AddYears(DateTime time, int years);

[VB] Overrides Public Function AddYears(ByVal time As DateTime, ByVal years As Integer) As DateTime

[JScript] public override function AddYears(time : DateTime, years : int) : DateTime;

Description

Returns a **System.DateTime** that is the specified number of years away from the specified **System.DateTime** .

Return Value: The **System.DateTime** that results from adding the specified number of years to the specified **System.DateTime** .

The day part of the resulting **System.DateTime** is affected if the resulting day is not a valid day in the resulting month of the resulting year; it is changed to the last valid day in the resulting month of the resulting year. The time-of-day part of the resulting **System.DateTime** remains the same as the specified **System.DateTime** . The **System.DateTime** instance to add. The number of years to add.

GetDayOfMonth

1
2 [C#] public override int GetDayOfMonth(DateTime time);

3 [C++] public: int GetDayOfMonth(DateTime time);

4 [VB] Overrides Public Function GetDayOfMonth(ByVal time As DateTime) As

5 Integer

6 [JScript] public override function GetDayOfMonth(time : DateTime) : int;

7
8 *Description*

9 Gets the day of the month in the specified **System.DateTime** .

10 *Return Value:* An integer from 1 to 31 that represents the day of the month in *time*

11 . The **System.DateTime** instance to read.

12 **GetDayOfWeek**

13
14 [C#] public override DayOfWeek GetDayOfWeek(DateTime time);

15 [C++] public: DayOfWeek GetDayOfWeek(DateTime time);

16 [VB] Overrides Public Function GetDayOfWeek(ByVal time As DateTime) As

17 DayOfWeek

18 [JScript] public override function GetDayOfWeek(time : DateTime) :

19 DayOfWeek;

20
21 *Description*

22 Gets the day of the week in the specified **System.DateTime** .

23 *Return Value:* A **System.DayOfWeek** value that represents the day of the week in

24 *time* .

The **System.DayOfWeek** values are Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, and Saturday. The **System.DateTime** instance to read.

GetDayOfYear

[C#] public override int GetDayOfYear(DateTime time);

[C++] public: int GetDayOfYear(DateTime time);

[VB] Overrides Public Function GetDayOfYear(ByVal time As DateTime) As Integer

[JScript] public override function GetDayOfYear(time : DateTime) : int;

Description

Gets the day of the year in the specified **System.DateTime** .

Return Value: An integer from 1 to 366 that represents the day of the year in *time* .

The **System.DateTime** instance to read.

GetDaysInMonth

[C#] public override int GetDaysInMonth(int year, int month, int era);

[C++] public: int GetDaysInMonth(int year, int month, int era);

[VB] Overrides Public Function GetDaysInMonth(ByVal year As Integer, ByVal month As Integer, ByVal era As Integer) As Integer

[JScript] public override function GetDaysInMonth(year : int, month : int, era : int) : int; Gets the number of days in the specified month.

Description

Gets the number of days in the month specified by the *year* , *month* , and *era* parameters.

Return Value: The number of days in the specified month in the specified year in the specified era.

For example, this method might return 28 or 29 for February (*month* = 2), depending on whether *year* is a leap year. An integer that represents the year. An integer that represents the month. An integer that represents the era.

GetDaysInYear

[C#] public override int GetDaysInYear(int year, int era);

[C++] public: int GetDaysInYear(int year, int era);

[VB] Overrides Public Function GetDaysInYear(ByVal year As Integer, ByVal era As Integer) As Integer

[JScript] public override function GetDaysInYear(year : int, era : int) : int; Gets the number of days in the specified year.

Description

Gets the number of days in the year specified by the *year* and *era* parameters.

Return Value: The number of days in the specified year in the specified era.

For example, this method might return 365 or 366, depending on whether *year* is a leap year. An integer that represents the year. An integer that represents the era.

GetEra

1
2 [C#] public override int GetEra(DateTime time);

3 [C++] public: int GetEra(DateTime time);

4 [VB] Overrides Public Function GetEra(ByVal time As DateTime) As Integer

5 [JScript] public override function GetEra(time : DateTime) : int;

6
7 *Description*

8 Gets the era in the specified **System.DateTime** .

9 *Return Value:* An integer that represents the era in *time* .

10 The Gregorian calendar recognizes two eras: B.C. (before Christ) or B.C.E.
11 (before common era), and A.D. (Latin "Anno Domini", which means "in the year
12 of the Lord") or C.E. (common era). This implementation of the
13 **System.Globalization.GregorianCalendar** class recognizes only the current era
14 (A.D. or C.E.). The **System.DateTime** instance to read.

15 **GetMonth**

16
17 [C#] public override int GetMonth(DateTime time);

18 [C++] public: int GetMonth(DateTime time);

19 [VB] Overrides Public Function GetMonth(ByVal time As DateTime) As Integer

20 [JScript] public override function GetMonth(time : DateTime) : int;

21
22 *Description*

23 Gets the month in the specified **System.DateTime** .

24 *Return Value:* An integer between 1 and 12 that represents the month in *time* . The
25 **System.DateTime** instance to read.

GetMonthsInYear

[C#] public override int GetMonthsInYear(int year, int era);

[C++] public: int GetMonthsInYear(int year, int era);

[VB] Overrides Public Function GetMonthsInYear(ByVal year As Integer, ByVal era As Integer) As Integer

[JScript] public override function GetMonthsInYear(year : int, era : int) : int; Gets the number of months in the specified year.

Description

Gets the number of months in the year specified by the *year* and *era* parameters.

Return Value: The number of months in the specified year in the specified era. An integer that represents the year. An integer that represents the era.

GetYear

[C#] public override int GetYear(DateTime time);

[C++] public: int GetYear(DateTime time);

[VB] Overrides Public Function GetYear(ByVal time As DateTime) As Integer

[JScript] public override function GetYear(time : DateTime) : int;

Description

Gets the year in the specified **System.DateTime** .

Return Value: An integer between 1 and 9999 that represents the year in *time* . The **System.DateTime** instance to read.

IsLeapDay

[C#] public override bool IsLeapDay(int year, int month, int day, int era);
[C++] public: bool IsLeapDay(int year, int month, int day, int era);
[VB] Overrides Public Function IsLeapDay(ByVal year As Integer, ByVal month As Integer, ByVal day As Integer, ByVal era As Integer) As Boolean
[JScript] public override function IsLeapDay(year : int, month : int, day : int, era : int) : Boolean; Determines whether the specified day is a leap day.

Description

Determines whether the date specified by the *year* , *month* , *day* , and *era* parameters is a leap day.

Return Value: **true** if the specified day is a leap day; otherwise, **false** .

A leap year in the Gregorian calendar is defined as a year that is evenly divisible by four, except if it is divisible by 100; however, years that are divisible by 400 are leap years. For example, the year 1900 was not a leap year, but the year 2000 was. A common year has 365 days and a leap year has 366 days. An integer that represents the year. An integer that represents the month. An integer that represents the day. An integer that represents the era.

IsLeapMonth

[C#] public override bool IsLeapMonth(int year, int month, int era);
[C++] public: bool IsLeapMonth(int year, int month, int era);
[VB] Overrides Public Function IsLeapMonth(ByVal year As Integer, ByVal month As Integer, ByVal era As Integer) As Boolean

1 [JScript] public override function IsLeapMonth(year : int, month : int, era : int) :

2 Boolean; Determines whether the specified month is a leap month.

3
4 *Description*

5 Determines whether the month specified by the *year* , *month* , and *era*
6 parameters is a leap month.

7 *Return Value:* This method always returns **false** , unless overridden by a derived
8 class.

9 A leap year in the Gregorian calendar is defined as a year that is evenly
10 divisible by four, except if it is divisible by 100; however, years that are divisible
11 by 400 are leap years. For example, the year 1900 was not a leap year, but the year
12 2000 was. A common year has 365 days and a leap year has 366 days. An integer
13 that represents the year. An integer that represents the month. An integer that
14 represents the era.

15 **IsLeapYear**

16
17 [C#] public override bool IsLeapYear(int year, int era);

18 [C++] public: bool IsLeapYear(int year, int era);

19 [VB] Overrides Public Function IsLeapYear(ByVal year As Integer, ByVal era As
20 Integer) As Boolean

21 [JScript] public override function IsLeapYear(year : int, era : int) : Boolean;

22 Determines whether the specified year is a leap year.

23
24 *Description*

Determines whether the year specified by the *year* and *era* parameters is a leap year.

Return Value: **true** if the specified year is a leap year; otherwise, **false** .

A leap year in the Gregorian calendar is defined as a year that is evenly divisible by four, except if it is divisible by 100; however, years that are divisible by 400 are leap years. For example, the year 1900 was not a leap year, but the year 2000 was. A common year has 365 days and a leap year has 366 days. An integer that represents the year. An integer that represents the era.

ToDateTime

[C#] public override DateTime.ToDateTime(int year, int month, int day, int hour, int minute, int second, int millisecond, int era);

[C++] public: DateTime ToDateTime(int year, int month, int day, int hour, int minute, int second, int millisecond, int era);

[VB] Overrides Public Function ToDateTime(ByVal year As Integer, ByVal month As Integer, ByVal day As Integer, ByVal hour As Integer, ByVal minute As Integer, ByVal second As Integer, ByVal millisecond As Integer, ByVal era As Integer) As DateTime

[JScript] public override function ToDateTime(year : int, month : int, day : int, hour : int, minute : int, second : int, millisecond : int, era : int) : DateTime; Returns a **System.DateTime** that is set to the specified date.

Description

Returns a **System.DateTime** that is set to the specified date and time in the specified era.

Return Value: The **System.DateTime** instance set to the specified date and time in the current era. An integer that represents the year. An integer that represents the month. An integer that represents the day. An integer that represents the hour. An integer that represents the minute. An integer that represents the second. An integer that represents the millisecond. An integer that represents the era.

ToFourDigitYear

[C#] public override int ToFourDigitYear(int year);

[C++] public: int ToFourDigitYear(int year);

[VB] Overrides Public Function ToFourDigitYear(ByVal year As Integer) As Integer

[JScript] public override function ToFourDigitYear(year : int) : int;

Description

Converts the specified two-digit year to a four-digit year by using the **System.Globalization.GregorianCalendar.TwoDigitYearMax** property to determine the appropriate century.

Return Value: An integer that contains the four-digit representation of *year*.

System.Globalization.GregorianCalendar.TwoDigitYearMax is the last year in the 100-year range that can be represented by a two-digit year. The century is determined by finding the sole occurrence of the two-digit *year* within that 100-year range. For example, if

System.Globalization.GregorianCalendar.TwoDigitYearMax is set to 2029, the 100-year range is from 1930 to 2029; therefore, a 2-digit value of 30 is

1 interpreted as 1930, while a 2-digit value of 29 is interpreted as 2029. A two-digit
2 integer that represents the year to convert.

3 GregorianCalendarTypes enumeration (System.Globalization)

4 ToString

5
6
7 *Description*

8 Defines the different language versions of the Gregorian calendar.

9 Each version differs by language.

10 ToString

11
12 [C#] public const GregorianCalendarTypes Arabic;

13 [C++] public: const GregorianCalendarTypes Arabic;

14 [VB] Public Const Arabic As GregorianCalendarTypes

15 [JScript] public var Arabic : GregorianCalendarTypes;

16
17 *Description*

18 Refers to the Arabic version of the Gregorian calendar.

19 ToString

20
21 [C#] public const GregorianCalendarTypes Localized;

22 [C++] public: const GregorianCalendarTypes Localized;

23 [VB] Public Const Localized As GregorianCalendarTypes

24 [JScript] public var Localized : GregorianCalendarTypes;

1
2 *Description*

3 Refers to the localized version of the Gregorian calendar, based on the
4 language of the **System.Globalization.CultureInfo** that uses the
5 **System.Globalization.DateTimeFormatInfo** instance.

6 ToString

7
8 [C#] public const GregorianCalendarTypes MiddleEastFrench;

9 [C++] public: const GregorianCalendarTypes MiddleEastFrench;

10 [VB] Public Const MiddleEastFrench As GregorianCalendarTypes

11 [JScript] public var MiddleEastFrench : GregorianCalendarTypes;

12
13 *Description*

14 Refers to the Middle East French version of the Gregorian calendar.

15 ToString

16
17 [C#] public const GregorianCalendarTypes TransliteratedEnglish;

18 [C++] public: const GregorianCalendarTypes TransliteratedEnglish;

19 [VB] Public Const TransliteratedEnglish As GregorianCalendarTypes

20 [JScript] public var TransliteratedEnglish : GregorianCalendarTypes;

21
22 *Description*

23 Refers to the transliterated English version of the Gregorian calendar.

24 ToString

1
2 [C#] public const GregorianCalendarTypes TransliteratedFrench;

3 [C++] public: const GregorianCalendarTypes TransliteratedFrench;

4 [VB] Public Const TransliteratedFrench As GregorianCalendarTypes

5 [JScript] public var TransliteratedFrench : GregorianCalendarTypes;

6
7 *Description*

8 Refers to the transliterated French version of the Gregorian calendar.

9 ToString

10
11 [C#] public const GregorianCalendarTypes USEnglish;

12 [C++] public: const GregorianCalendarTypes USEnglish;

13 [VB] Public Const USEnglish As GregorianCalendarTypes

14 [JScript] public var USEnglish : GregorianCalendarTypes;

15
16 *Description*

17 Refers to the US English version of the Gregorian calendar.

18 HebrewCalendar class (System.Globalization)

19 ToString

20
21
22 *Description*

23 Represents the Hebrew calendar.

24 The Hebrew calendar recognizes two eras: B.C.E. (before the common era)
25 and A.M. (Latin "Anno Mundi", which means "the year of the world"). This

1 implementation of the **System.Globalization.HebrewCalendar** class recognizes
2 only the current era (A.M.) and only the Hebrew years 5343 to 6000 (1582 to 2240
3 in the Gregorian calendar).

4 ToString

5
6 [C#] public static readonly int HebrewEra;

7 [C++] public: static int HebrewEra;

8 [VB] Public Shared ReadOnly HebrewEra As Integer

9 [JScript] public static var HebrewEra : int;

10
11 *Description*

12 Represents the current era.

13 The Hebrew calendar recognizes two eras: B.C.E. (before the common era)
14 and A.M. (Latin "Anno Mundi", which means "the year of the world"). This
15 implementation of the **System.Globalization.HebrewCalendar** class recognizes
16 only the current era (A.M.). This field always returns 1.

17 HebrewCalendar

18 *Example Syntax:*

19 ToString

20
21 [C#] public HebrewCalendar();

22 [C++] public: HebrewCalendar();

23 [VB] Public Sub New()

24 [JScript] public function HebrewCalendar();

1
2 *Description*

3 Initializes a new instance of the **System.Globalization.HebrewCalendar**
4 class.

5 Eras

6 ToString

7
8 [C#] public override int[] Eras {get;}

9 [C++] public: __property virtual int get_Eras();

10 [VB] Overrides Public ReadOnly Property Eras As Integer ()

11 [JScript] public function get Eras() : int[];

12
13 *Description*

14 Gets the list of eras in the **System.Globalization.HebrewCalendar** .

15 The Hebrew calendar recognizes two eras: B.C.E. (before the common era)
16 and A.M. (Latin "Anno Mundi", which means "the year of the world"). This
17 implementation of the **System.Globalization.HebrewCalendar** class recognizes
18 only the current era (A.M.). This property always returns an array with only one
19 element.

20 TwoDigitYearMax

21 ToString

22
23 [C#] public override int TwoDigitYearMax {get; set;}

24 [C++] public: __property virtual int get_TwoDigitYearMax();public: __property
25 virtual void set_TwoDigitYearMax(int);

1 [VB] Overrides Public Property TwoDigitYearMax As Integer

2 [JScript] public function get TwoDigitYearMax() : int; public function set

3 TwoDigitYearMax(int);

4
5 *Description*

6 Gets or sets the last year of a 100-year range that can be represented by a 2-
7 digit year.

8 This implementation of the **System.Globalization.HebrewCalendar** class
9 recognizes only the Hebrew years 5343 to 6000 (1582 to 2240 in the Gregorian
10 calendar).

11 **AddMonths**

12
13 [C#] public override DateTime AddMonths(DateTime time, int months);

14 [C++] public: DateTime AddMonths(DateTime time, int months);

15 [VB] Overrides Public Function AddMonths(ByVal time As DateTime, ByVal
16 months As Integer) As DateTime

17 [JScript] public override function AddMonths(time : DateTime, months : int) :
18 DateTime;

19
20 *Description*

21 Returns a **System.DateTime** that is the specified number of months away
22 from the specified **System.DateTime** .

23 *Return Value:* The **System.DateTime** that results from adding the specified
24 number of months to the specified **System.DateTime** .
25

This implementation of the **System.Globalization.HebrewCalendar** class recognizes only the Hebrew years 5343 to 6000 (1582 to 2240 in the Gregorian calendar). The **System.DateTime** instance to add. The number of months to add.

AddYears

[C#] public override DateTime AddYears(DateTime time, int years);

[C++] public: DateTime AddYears(DateTime time, int years);

[VB] Overrides Public Function AddYears(ByVal time As DateTime, ByVal years As Integer) As DateTime

[JScript] public override function AddYears(time : DateTime, years : int) : DateTime;

Description

Returns a **System.DateTime** that is the specified number of years away from the specified **System.DateTime**.

Return Value: The **System.DateTime** that results from adding the specified number of years to the specified **System.DateTime**.

This implementation of the **System.Globalization.HebrewCalendar** class recognizes only the Hebrew years 5343 to 6000 (1582 to 2240 in the Gregorian calendar). The **System.DateTime** instance to add. The number of years to add.

GetDayOfMonth

[C#] public override int GetDayOfMonth(DateTime time);

[C++] public: int GetDayOfMonth(DateTime time);

[VB] Overrides Public Function GetDayOfMonth(ByVal time As DateTime) As

Integer

[JScript] public override function GetDayOfMonth(time : DateTime) : int;

Description

Gets the day of the month in the specified **System.DateTime** .

Return Value: An integer from 1 to 30 that represents the day of the month in *time*

. The **System.DateTime** instance to read.

GetDayOfWeek

[C#] public override DayOfWeek GetDayOfWeek(DateTime time);

[C++] public: DayOfWeek GetDayOfWeek(DateTime time);

[VB] Overrides Public Function GetDayOfWeek(ByVal time As DateTime) As

DayOfWeek

[JScript] public override function GetDayOfWeek(time : DateTime) :

DayOfWeek;

Description

Gets the day of the week in the specified **System.DateTime** .

Return Value: A **System.DayOfWeek** value that represents the day of the week in *time* .

The **System.DayOfWeek** values are Sunday which indicates Yom Rishon, Monday which indicates Yom Sheni, Tuesday which indicates Yom Shlishi, Wednesday which indicates Yom Revicee, Thursday which indicates Yom Chamishi, Friday which indicates Yom Shishi, and Saturday which indicates Shabat. The **System.DateTime** instance to read.

GetDayOfYear

[C#] public override int GetDayOfYear(DateTime time);

[C++] public: int GetDayOfYear(DateTime time);

[VB] Overrides Public Function GetDayOfYear(ByVal time As DateTime) As Integer

[JScript] public override function GetDayOfYear(time : DateTime) : int;

Description

Gets the day of the year in the specified **System.DateTime**.

Return Value: An integer from 1 to 385 that represents the day of the year in *time*.

This implementation of the **System.Globalization.HebrewCalendar** class recognizes only the Hebrew years 5343 to 6000 (1582 to 2240 in the Gregorian calendar). The **System.DateTime** instance to read.

GetDaysInMonth

[C#] public override int GetDaysInMonth(int year, int month, int era);

[C++] public: int GetDaysInMonth(int year, int month, int era);

[VB] Overrides Public Function GetDaysInMonth(ByVal year As Integer, ByVal month As Integer, ByVal era As Integer) As Integer

[JScript] public override function GetDaysInMonth(year : int, month : int, era : int) : int; Gets the number of days in the specified month.

Description

Gets the number of days in the month specified by the *year* , *month* , and *era* parameters.

Return Value: The number of days in the specified month in the specified year in the specified era.

For example, this method might return 29 or 30 for Cheshvan, depending on the placement of Jewish holidays. An integer that represents the year. An integer that represents the month. An integer that represents the era.

GetDaysInYear

[C#] public override int GetDaysInYear(int year, int era);

[C++] public: int GetDaysInYear(int year, int era);

[VB] Overrides Public Function GetDaysInYear(ByVal year As Integer, ByVal era As Integer) As Integer

[JScript] public override function GetDaysInYear(year : int, era : int) : int; Gets the number of days in the specified year.

Description

Gets the number of days in the year specified by the *year* and *era* parameters.

Return Value: The number of days in the specified year in the specified era.

For example, this method might return an integer from 353 to 355 or from 383 to 385, depending on the placement of Jewish holidays and depending on whether *year* is a leap year. An integer that represents the year. An integer that represents the era.

GetEra

[C#] public override int GetEra(DateTime time);

[C++] public: int GetEra(DateTime time);

[VB] Overrides Public Function GetEra(ByVal time As DateTime) As Integer

[JScript] public override function GetEra(time : DateTime) : int;

Description

Gets the era in the specified **System.DateTime** .

Return Value: An integer that represents the era in *time* .

The Hebrew calendar recognizes two eras: B.C.E. (before the common era) and A.M. (Latin "Anno Mundi", which means "the year of the world"). This implementation of the **System.Globalization.HebrewCalendar** class recognizes only the current era (A.M.) and only the Hebrew years 5343 to 6000 (1582 to 2240 in the Gregorian calendar). The **System.DateTime** instance to read.

GetMonth

[C#] public override int GetMonth(DateTime time);

[C++] public: int GetMonth(DateTime time);

[VB] Overrides Public Function GetMonth(ByVal time As DateTime) As Integer

[JScript] public override function GetMonth(time : DateTime) : int;

Description

Gets the month in the specified **System.DateTime** .

Return Value: An integer between 1 and 13 that represents the month in *time* . The **System.DateTime** instance to read.

GetMonthsInYear

[C#] public override int GetMonthsInYear(int year, int era);

[C++] public: int GetMonthsInYear(int year, int era);

[VB] Overrides Public Function GetMonthsInYear(ByVal year As Integer, ByVal era As Integer) As Integer

[JScript] public override function GetMonthsInYear(year : int, era : int) : int; Gets the number of months in the specified year.

Description

Gets the number of months in the year specified by the *year* and *era* parameters.

Return Value: The number of months in the specified year in the specified era.

For example, this method might return 12 or 13, depending on whether *year* is a leap year. An integer that represents the year. An integer that represents the era.

GetYear

[C#] public override int GetYear(DateTime time);

[C++] public: int GetYear(DateTime time);

[VB] Overrides Public Function GetYear(ByVal time As DateTime) As Integer

[JScript] public override function GetYear(time : DateTime) : int;

Description

Gets the year in the specified **System.DateTime** .

Return Value: An integer between 1 and 9999 that represents the year in *time* .

This implementation of the **System.Globalization.HebrewCalendar** class recognizes only the Hebrew years 5343 to 6000 (1582 to 2240 in the Gregorian calendar). The **System.DateTime** instance to read.

IsLeapDay

[C#] public override bool IsLeapDay(int year, int month, int day, int era);

[C++] public: bool IsLeapDay(int year, int month, int day, int era);

[VB] Overrides Public Function IsLeapDay(ByVal year As Integer, ByVal month As Integer, ByVal day As Integer, ByVal era As Integer) As Boolean

[JScript] public override function IsLeapDay(year : int, month : int, day : int, era : int) : Boolean; Determines whether the specified day is a leap day.

Description

Determines whether the date specified by the *year* , *month* , *day* , and *era* parameters is a leap day.

Return Value: **true** if the specified day is a leap day; otherwise, **false** .

This implementation of the **System.Globalization.HebrewCalendar** class recognizes only the Hebrew years 5343 to 6000 (1582 to 2240 in the Gregorian calendar). An integer that represents the year. An integer that represents the month. An integer that represents the day. An integer that represents the era.

IsLeapMonth

[C#] public override bool IsLeapMonth(int year, int month, int era);

1 [C++] public: bool IsLeapMonth(int year, int month, int era);
2 [VB] Overrides Public Function IsLeapMonth(ByVal year As Integer, ByVal
3 month As Integer, ByVal era As Integer) As Boolean
4 [JScript] public override function IsLeapMonth(year : int, month : int, era : int) :
5 Boolean; Determines whether the specified month is a leap month.
6

7 *Description*

8 Determines whether the month specified by the *year* , *month* , and *era*
9 parameters is a leap month.

10 *Return Value:* **true** if the specified month is a leap month; otherwise, **false** .

11 This implementation of the **System.Globalization.HebrewCalendar** class
12 recognizes only the Hebrew years 5343 to 6000 (1582 to 2240 in the Gregorian
13 calendar). An integer that represents the year. An integer that represents the
14 month. An integer that represents the era.

15 **IsLeapYear**

16
17 [C#] public override bool IsLeapYear(int year, int era);
18 [C++] public: bool IsLeapYear(int year, int era);
19 [VB] Overrides Public Function IsLeapYear(ByVal year As Integer, ByVal era As
20 Integer) As Boolean
21 [JScript] public override function IsLeapYear(year : int, era : int) : Boolean;
22 Determines whether the specified year is a leap year.
23

24 *Description*

25

Determines whether the year specified by the *year* and *era* parameters is a leap year.

Return Value: **true** if the specified year is a leap year; otherwise, **false** .

This implementation of the **System.Globalization.HebrewCalendar** class recognizes only the Hebrew years 5343 to 6000 (1582 to 2240 in the Gregorian calendar). An integer that represents the year. An integer that represents the era.

ToDateTime

[C#] public override DateTime.ToDateTime(int year, int month, int day, int hour, int minute, int second, int millisecond, int era);

[C++] public: DateTime ToDateTime(int year, int month, int day, int hour, int minute, int second, int millisecond, int era);

[VB] Overrides Public Function ToDateTime(ByVal year As Integer, ByVal month As Integer, ByVal day As Integer, ByVal hour As Integer, ByVal minute As Integer, ByVal second As Integer, ByVal millisecond As Integer, ByVal era As Integer) As DateTime

[JScript] public override function ToDateTime(year : int, month : int, day : int, hour : int, minute : int, second : int, millisecond : int, era : int) : DateTime; Returns a **System.DateTime** that is set to the specified date.

Description

Returns a **System.DateTime** that is set to the specified date and time in the specified era.

Return Value: The **System.DateTime** instance set to the specified date and time in the current era.

This implementation of the **System.Globalization.HebrewCalendar** class recognizes only the Hebrew years 5343 to 6000 (1582 to 2240 in the Gregorian calendar). An integer that represents the year. An integer that represents the month. An integer that represents the day. An integer that represents the hour. An integer that represents the minute. An integer that represents the second. An integer that represents the millisecond. An integer that represents the era.

ToFourDigitYear

[C#] public override int ToFourDigitYear(int year);

[C++] public: int ToFourDigitYear(int year);

[VB] Overrides Public Function ToFourDigitYear(ByVal year As Integer) As Integer

[JScript] public override function ToFourDigitYear(year : int) : int;

Description

Converts the specified two-digit year to a four-digit year by using the **System.Globalization.HebrewCalendar.TwoDigitYearMax** property to determine the appropriate century.

Return Value: An integer that contains the four-digit representation of *year*.

This implementation of the **System.Globalization.HebrewCalendar** class recognizes only the Hebrew years 5343 to 6000 (1582 to 2240 in the Gregorian calendar). A two-digit integer that represents the year to convert.

HijriCalendar class (System.Globalization)

ToString

1
2
3 *Description*

4 Represents the Hijri calendar.

5 The Hijri calendar recognizes one era: A.H. (Latin "Anno Hegirae", which
6 means "the year of the migration" in reference to the migration of Muhammad
7 from Mecca).

8 ToString

9
10 [C#] public static readonly int HijriEra;

11 [C++] public: static int HijriEra;

12 [VB] Public Shared ReadOnly HijriEra As Integer

13 [JScript] public static var HijriEra : int;

14
15 *Description*

16 Represents the current era.

17 The Hijri calendar recognizes one era: A.H. (Latin "Anno Hegirae", which
18 means "the year of the migration" referring to the migration of Muhammad from
19 Mecca).

20 HijriCalendar

21 *Example Syntax:*

22 ToString

23
24 [C#] public HijriCalendar();

25 [C++] public: HijriCalendar();

1 [VB] Public Sub New()

2 [JScript] public function HijriCalendar();

3
4 *Description*

5 Initializes a new instance of the **System.Globalization.HijriCalendar**
6 class.

7 Eras

8 ToString

9
10 [C#] public override int[] Eras {get;}

11 [C++] public: __property virtual int get_Eras();

12 [VB] Overrides Public ReadOnly Property Eras As Integer ()

13 [JScript] public function get Eras() : int[];

14
15 *Description*

16 Gets the list of eras in the **System.Globalization.HijriCalendar** .

17 The Hijri calendar recognizes one era: A.H. (Latin "Anno Hegirae", which
18 means "the year of the migration" referring to the migration of Muhammad from
19 Mecca).

20 TwoDigitYearMax

21 ToString

22
23 [C#] public override int TwoDigitYearMax {get; set;}

24 [C++] public: __property virtual int get_TwoDigitYearMax();public: __property

25 virtual void set_TwoDigitYearMax(int);

1 [VB] Overrides Public Property TwoDigitYearMax As Integer

2 [JScript] public function get TwoDigitYearMax() : int;public function set

3 TwoDigitYearMax(int);

4
5 *Description*

6 Gets or sets the last year of a 100-year range that can be represented by a 2-
7 digit year.

8 This property allows a 2-digit year to be properly translated to a 4-digit
9 year. For example, if this property is set to 1429, the 100-year range is from 1330
10 to 1429; therefore, a 2-digit value of 30 is interpreted as 1330, while a 2-digit
11 value of 29 is interpreted as 1429.

12 *AddMonths*

13
14 [C#] public override DateTime AddMonths(DateTime time, int months);

15 [C++] public: DateTime AddMonths(DateTime time, int months);

16 [VB] Overrides Public Function AddMonths(ByVal time As DateTime, ByVal
17 months As Integer) As DateTime

18 [JScript] public override function AddMonths(time : DateTime, months : int) :

19 DateTime;

20
21 *Description*

22 Returns a **System.DateTime** that is the specified number of months away
23 from the specified **System.DateTime** .

24 *Return Value:* The **System.DateTime** that results from adding the specified
25 number of months to the specified **System.DateTime** .

The year part of the resulting **System.DateTime** is affected if the resulting month is beyond the last month of the current year. The day part of the resulting **System.DateTime** is also affected if the resulting day is not a valid day in the resulting month of the resulting year; it is changed to the last valid day in the resulting month of the resulting year. The time-of-day part of the resulting **System.DateTime** remains the same as the specified **System.DateTime** . The **System.DateTime** instance to add. The number of months to add.

AddYears

[C#] public override DateTime AddYears(DateTime time, int years);

[C++] public: DateTime AddYears(DateTime time, int years);

[VB] Overrides Public Function AddYears(ByVal time As DateTime, ByVal years As Integer) As DateTime

[JScript] public override function AddYears(time : DateTime, years : int) : DateTime;

Description

Returns a **System.DateTime** that is the specified number of years away from the specified **System.DateTime** .

Return Value: The **System.DateTime** that results from adding the specified number of years to the specified **System.DateTime** .

The day part of the resulting **System.DateTime** is affected if the resulting day is not a valid day in the resulting month of the resulting year; it is changed to the last valid day in the resulting month of the resulting year. The time-of-day part of the resulting **System.DateTime** remains the same as the specified

System.DateTime . The **System.DateTime** instance to add. The number of years to add.

GetDayOfMonth

[C#] public override int GetDayOfMonth(DateTime time);

[C++] public: int GetDayOfMonth(DateTime time);

[VB] Overrides Public Function GetDayOfMonth(ByVal time As DateTime) As

Integer

[JScript] public override function GetDayOfMonth(time : DateTime) : int;

Description

Gets the day of the month in the specified **System.DateTime** .

Return Value: An integer from 1 to 30 that represents the day of the month in *time* . The **System.DateTime** instance to read.

GetDayOfWeek

[C#] public override DayOfWeek GetDayOfWeek(DateTime time);

[C++] public: DayOfWeek GetDayOfWeek(DateTime time);

[VB] Overrides Public Function GetDayOfWeek(ByVal time As DateTime) As

DayOfWeek

[JScript] public override function GetDayOfWeek(time : DateTime) :

DayOfWeek;

Description

Gets the day of the week in the specified **System.DateTime** .

Return Value: A **System.DayOfWeek** value that represents the day of the week in *time* .

The **System.DayOfWeek** values are Sunday which indicates Al-Ahad, Monday which indicates Al-Ithnayn, Tuesday which indicates At-Thulaathaa', Wednesday which indicates Al-Arbi'aa', Thursday which indicates Al-Khamiis, Friday which indicates Al-Jumu'ah, and Saturday which indicates As-Sabt. The **System.DateTime** instance to read.

GetDayOfYear

[C#] public override int GetDayOfYear(DateTime time);

[C++] public: int GetDayOfYear(DateTime time);

[VB] Overrides Public Function GetDayOfYear(ByVal time As DateTime) As Integer

[JScript] public override function GetDayOfYear(time : DateTime) : int;

Description

Gets the day of the year in the specified **System.DateTime** .

Return Value: An integer from 1 to 355 that represents the day of the year in *time* .

The **System.DateTime** instance to read.

GetDaysInMonth

[C#] public override int GetDaysInMonth(int year, int month, int era);

[C++] public: int GetDaysInMonth(int year, int month, int era);

[VB] Overrides Public Function GetDaysInMonth(ByVal year As Integer, ByVal

1 month As Integer, ByVal era As Integer) As Integer

2 [JScript] public override function GetDaysInMonth(year : int, month : int, era :
3 int) : int; Gets the number of days in the specified month.

5 *Description*

6 Gets the number of days in the month specified by the *year* , *month* , and
7 *era* parameters.

8 *Return Value:* The number of days in the specified month in the specified year in
9 the specified era.

10 For example, this method might return 29 or 30 for Zulhijjah (*month* = 12),
11 depending on whether *year* is a leap year. An integer that represents the year. An
12 integer that represents the month. An integer that represents the era.

13 *GetDaysInYear*

14
15 [C#] public override int GetDaysInYear(int year, int era);

16 [C++] public: int GetDaysInYear(int year, int era);

17 [VB] Overrides Public Function GetDaysInYear(ByVal year As Integer, ByVal
18 era As Integer) As Integer

19 [JScript] public override function GetDaysInYear(year : int, era : int) : int; Gets
20 the number of days in the specified year.

22 *Description*

23 Gets the number of days in the year specified by the *year* and *era*
24 parameters.

25 *Return Value:* The number of days in the specified year in the specified era.

For example, this method might return 354 or 355, depending on whether *year* is a leap year. An integer that represents the year. An integer that represents the era.

GetEra

[C#] public override int GetEra(DateTime time);

[C++] public: int GetEra(DateTime time);

[VB] Overrides Public Function GetEra(ByVal time As DateTime) As Integer

[JScript] public override function GetEra(time : DateTime) : int;

Description

Gets the era in the specified **System.DateTime** .

Return Value: An integer that represents the era in *time* .

The Hijri calendar recognizes one era: A.H. (Latin "Anno Hegirae", which means "the year of the migration" referring to the migration of Muhammad from Mecca). The **System.DateTime** instance to read.

GetMonth

[C#] public override int GetMonth(DateTime time);

[C++] public: int GetMonth(DateTime time);

[VB] Overrides Public Function GetMonth(ByVal time As DateTime) As Integer

[JScript] public override function GetMonth(time : DateTime) : int;

Description

Gets the month in the specified **System.DateTime** .

Return Value: An integer between 1 and 12 that represents the month in *time* . The **System.DateTime** instance to read.

GetMonthsInYear

[C#] public override int GetMonthsInYear(int year, int era);

[C++] public: int GetMonthsInYear(int year, int era);

[VB] Overrides Public Function GetMonthsInYear(ByVal year As Integer, ByVal era As Integer) As Integer

[JScript] public override function GetMonthsInYear(year : int, era : int) : int; Gets the number of months in the specified year.

Description

Gets the number of months in the year specified by the *year* and *era* parameters.

Return Value: The number of months in the specified year in the specified era. An integer that represents the year. An integer that represents the era.

GetYear

[C#] public override int GetYear(DateTime time);

[C++] public: int GetYear(DateTime time);

[VB] Overrides Public Function GetYear(ByVal time As DateTime) As Integer

[JScript] public override function GetYear(time : DateTime) : int;

Description

Gets the year in the specified **System.DateTime** .

Return Value: An integer between 1 and 9999 that represents the year in *time* . The **System.DateTime** instance to read.

IsLeapDay

[C#] public override bool IsLeapDay(int year, int month, int day, int era);

[C++] public: bool IsLeapDay(int year, int month, int day, int era);

[VB] Overrides Public Function IsLeapDay(ByVal year As Integer, ByVal month As Integer, ByVal day As Integer, ByVal era As Integer) As Boolean

[JScript] public override function IsLeapDay(year : int, month : int, day : int, era : int) : Boolean; Determines whether the specified day is a leap day.

Description

Determines whether the date specified by the *year* , *month* , *day* , and *era* parameters is a leap day.

Return Value: **true** if the specified day is a leap day; otherwise, **false** .

In every 30-year cycle that ends with a year that is evenly divisible by 30, the 2nd, 5th, 7th, 10th, 13th, 16th, 18th, 21st, 24th, 26th, and 29th years are leap years. A common year has 354 days and a leap year has 355 days. An integer that represents the year. An integer that represents the month. An integer that represents the day. An integer that represents the era.

IsLeapMonth

[C#] public override bool IsLeapMonth(int year, int month, int era);

[C++] public: bool IsLeapMonth(int year, int month, int era);

[VB] Overrides Public Function IsLeapMonth(ByVal year As Integer, ByVal month As Integer, ByVal era As Integer) As Boolean

[JScript] public override function IsLeapMonth(year : int, month : int, era : int) : Boolean; Determines whether the specified month is a leap month.

Description

Determines whether the month specified by the *year* , *month* , and *era* parameters is a leap month.

Return Value: This method always returns **false** , unless overridden by a derived class.

In every 30-year cycle that ends with a year that is evenly divisible by 30, the 2nd, 5th, 7th, 10th, 13th, 16th, 18th, 21st, 24th, 26th, and 29th years are leap years. A common year has 354 days and a leap year has 355 days. An integer that represents the year. An integer that represents the month. An integer that represents the era.

IsLeapYear

[C#] public override bool IsLeapYear(int year, int era);

[C++] public: bool IsLeapYear(int year, int era);

[VB] Overrides Public Function IsLeapYear(ByVal year As Integer, ByVal era As Integer) As Boolean

[JScript] public override function IsLeapYear(year : int, era : int) : Boolean;

Determines whether the specified year is a leap year.

Description

Determines whether the year specified by the *year* and *era* parameters is a leap year.

Return Value: **true** if the specified year is a leap year; otherwise, **false** .

In every 30-year cycle that ends with a year that is evenly divisible by 30, the 2nd, 5th, 7th, 10th, 13th, 16th, 18th, 21st, 24th, 26th, and 29th years are leap years. A common year has 354 days and a leap year has 355 days. An integer that represents the year. An integer that represents the era.

ToDateTime

[C#] public override DateTime.ToDateTime(int year, int month, int day, int hour, int minute, int second, int millisecond, int era);

[C++] public: DateTime ToDateTime(int year, int month, int day, int hour, int minute, int second, int millisecond, int era);

[VB] Overrides Public Function ToDateTime(ByVal year As Integer, ByVal month As Integer, ByVal day As Integer, ByVal hour As Integer, ByVal minute As Integer, ByVal second As Integer, ByVal millisecond As Integer, ByVal era As Integer) As DateTime

[JScript] public override function ToDateTime(year : int, month : int, day : int, hour : int, minute : int, second : int, millisecond : int, era : int) : DateTime; Returns a **System.DateTime** that is set to the specified date.

Description

Returns a **System.DateTime** that is set to the specified date and time in the specified era.

Return Value: The **System.DateTime** instance set to the specified date and time in

the current era. An integer that represents the year. An integer that represents the month. An integer that represents the day. An integer that represents the hour. An integer that represents the minute. An integer that represents the second. An integer that represents the millisecond. An integer that represents the era.

ToFourDigitYear

[C#] public override int ToFourDigitYear(int year);

[C++] public: int ToFourDigitYear(int year);

[VB] Overrides Public Function ToFourDigitYear(ByVal year As Integer) As

Integer

[JScript] public override function ToFourDigitYear(year : int) : int;

Description

Converts the specified two-digit year to a four-digit year by using the **System.Globalization.HijriCalendar.TwoDigitYearMax** property to determine the appropriate century.

Return Value: An integer that contains the four-digit representation of *year*.

System.Globalization.HijriCalendar.TwoDigitYearMax is the last year in the 100-year range that can be represented by a two-digit year. The century is determined by finding the sole occurrence of the two-digit *year* within that 100-year range. For example, if **System.Globalization.HijriCalendar.TwoDigitYearMax** is set to 1429, the 100-year range is from 1330 to 1429; therefore, a 2-digit value of 30 is interpreted as 1330, while a 2-digit value of 29 is interpreted as 1429. A two-digit integer that represents the year to convert.

JapaneseCalendar class (System.Globalization)

ToString

Description

Represents the Japanese calendar.

The Japanese calendar, which is also known as the Wareki calendar, works exactly like the Gregorian calendar, except that the year and era are different.

JapaneseCalendar

Example Syntax:

ToString

[C#] public JapaneseCalendar();

[C++] public: JapaneseCalendar();

[VB] Public Sub New()

[JScript] public function JapaneseCalendar();

Description

Initializes a new instance of the **System.Globalization.JapaneseCalendar** class.

Eras

ToString

[C#] public override int[] Eras {get;}

[C++] public: __property virtual int get_Eras();

1 [VB] Overrides Public ReadOnly Property Eras As Integer ()

2 [JScript] public function get Eras() : int[];

3
4 *Description*

5 Gets the list of eras in the **System.Globalization.JapaneseCalendar** .

6 The Japanese calendar recognizes one era for every emperor's reign. The
7 current era is the Heisei era, which began in the Gregorian calendar year 1989.
8 The era name is typically displayed before the year. For example, the Gregorian
9 calendar year 2001 is the Wareki calendar year Heisei 13. Note that the first year
10 of an era is called "Gannen"; therefore, the Gregorian calendar year 1989 was the
11 Wareki calendar year Heisei Gannen.

12 TwoDigitYearMax

13 ToString

14
15 [C#] public override int TwoDigitYearMax {get; set;}

16 [C++] public: __property virtual int get_TwoDigitYearMax();public: __property
17 virtual void set_TwoDigitYearMax(int);

18 [VB] Overrides Public Property TwoDigitYearMax As Integer

19 [JScript] public function get TwoDigitYearMax() : int;public function set
20 TwoDigitYearMax(int);

21
22 *Description*

23 Gets or sets the last year of a 100-year range that can be represented by a 2-
24 digit year.

This property implements

System.Globalization.Calendar.TwoDigitYearMax .

AddMonths

[C#] public override DateTime AddMonths(DateTime time, int months);

[C++] public: DateTime AddMonths(DateTime time, int months);

[VB] Overrides Public Function AddMonths(ByVal time As DateTime, ByVal months As Integer) As DateTime

[JScript] public override function AddMonths(time : DateTime, months : int) : DateTime;

Description

Returns a **System.DateTime** that is the specified number of months away from the specified **System.DateTime** .

Return Value: The **System.DateTime** that results from adding the specified number of months to the specified **System.DateTime** .

The year part of the resulting **System.DateTime** is affected if the resulting month is beyond the last month of the current year. The day part of the resulting **System.DateTime** is also affected if the resulting day is not a valid day in the resulting month of the resulting year; it is changed to the last valid day in the resulting month of the resulting year. The time-of-day part of the resulting **System.DateTime** remains the same as the specified **System.DateTime** . The **System.DateTime** instance to add. The number of months to add.

AddYears

1
2 [C#] public override DateTime AddYears(DateTime time, int years);

3 [C++] public: DateTime AddYears(DateTime time, int years);

4 [VB] Overrides Public Function AddYears(ByVal time As DateTime, ByVal years
5 As Integer) As DateTime

6 [JScript] public override function AddYears(time : DateTime, years : int) :
7 DateTime;

8
9 *Description*

10 Returns a **System.DateTime** that is the specified number of years away
11 from the specified **System.DateTime** .

12 *Return Value:* The **System.DateTime** that results from adding the specified
13 number of years to the specified **System.DateTime** .

14 The day part of the resulting **System.DateTime** is affected if the resulting
15 day is not a valid day in the resulting month of the resulting year; it is changed to
16 the last valid day in the resulting month of the resulting year. The time-of-day part
17 of the resulting **System.DateTime** remains the same as the specified
18 **System.DateTime** . The **System.DateTime** instance to add. The number of years
19 to add.

20 **GetDayOfMonth**

21
22 [C#] public override int GetDayOfMonth(DateTime time);

23 [C++] public: int GetDayOfMonth(DateTime time);

24 [VB] Overrides Public Function GetDayOfMonth(ByVal time As DateTime) As
25 Integer

1 [JScript] public override function GetDayOfMonth(time : DateTime) : int;

3 *Description*

4 Gets the day of the month in the specified **System.DateTime** .

5 *Return Value:* An integer from 1 to 31 that represents the day of the month in *time*

6 . The **System.DateTime** instance to read.

7 GetDayOfWeek

9 [C#] public override DayOfWeek GetDayOfWeek(DateTime time);

10 [C++] public: DayOfWeek GetDayOfWeek(DateTime time);

11 [VB] Overrides Public Function GetDayOfWeek(ByVal time As DateTime) As

12 DayOfWeek

13 [JScript] public override function GetDayOfWeek(time : DateTime) :

14 DayOfWeek;

16 *Description*

17 Gets the day of the week in the specified **System.DateTime** .

18 *Return Value:* A **System.DayOfWeek** value that represents the day of the week in
19 *time* .

20 The **System.DayOfWeek** values are Sunday which indicates NichiYoubi,
21 Monday which indicates GetsuYoubi, Tuesday which indicates KaYoubi,
22 Wednesday which indicates SuiYoubi, Thursday which indicates MokuYoubi,
23 Friday which indicates KinYoubi, and Saturday which indicates DouYoubi. The
24 **System.DateTime** instance to read.

25 GetDayOfYear

1
2 [C#] public override int GetDayOfYear(DateTime time);

3 [C++] public: int GetDayOfYear(DateTime time);

4 [VB] Overrides Public Function GetDayOfYear(ByVal time As DateTime) As
5 Integer

6 [JScript] public override function GetDayOfYear(time : DateTime) : int;

7
8 *Description*

9 Gets the day of the year in the specified **System.DateTime** .

10 *Return Value:* An integer from 1 to 366 that represents the day of the year in *time* .

11 The **System.DateTime** instance to read.

12 **GetDaysInMonth**

13
14 [C#] public override int GetDaysInMonth(int year, int month, int era);

15 [C++] public: int GetDaysInMonth(int year, int month, int era);

16 [VB] Overrides Public Function GetDaysInMonth(ByVal year As Integer, ByVal
17 month As Integer, ByVal era As Integer) As Integer

18 [JScript] public override function GetDaysInMonth(year : int, month : int, era :
19 int) : int; Gets the number of days in the specified month.

20
21 *Description*

22 Gets the number of days in the month specified by the *year* , *month* , and
23 *era* parameters.

24 *Return Value:* The number of days in the specified month in the specified year in
25 the specified era.

For example, this method might return 28 or 29 for NiGatsu (February, *month* = 2), depending on whether *year* is a leap year. An integer that represents the year. An integer that represents the month. An integer that represents the era.

GetDaysInYear

[C#] public override int GetDaysInYear(int year, int era);

[C++] public: int GetDaysInYear(int year, int era);

[VB] Overrides Public Function GetDaysInYear(ByVal year As Integer, ByVal era As Integer) As Integer

[JScript] public override function GetDaysInYear(year : int, era : int) : int; Gets the number of days in the specified year.

Description

Gets the number of days in the year specified by the *year* and *era* parameters.

Return Value: The number of days in the specified year in the specified era.

For example, this method might return 365 or 366, depending on whether *year* is a leap year. An integer that represents the year. An integer that represents the era.

GetEra

[C#] public override int GetEra(DateTime time);

[C++] public: int GetEra(DateTime time);

[VB] Overrides Public Function GetEra(ByVal time As DateTime) As Integer

[JScript] public override function GetEra(time : DateTime) : int;

Description

Gets the era in the specified **System.DateTime** .

Return Value: An integer that represents the era in *time* .

The Japanese calendar recognizes one era for every emperor's reign. The current era is the Heisei era, which began in the Gregorian calendar year 1989. The era name is typically displayed before the year. For example, the Gregorian calendar year 2001 is the Wareki calendar year Heisei 13. Note that the first year of an era is called "Gannen"; therefore, the Gregorian calendar year 1989 was the Wareki calendar year Heisei Gannen. The **System.DateTime** instance to read.

GetMonth

[C#] public override int GetMonth(DateTime time);

[C++] public: int GetMonth(DateTime time);

[VB] Overrides Public Function GetMonth(ByVal time As DateTime) As Integer

[JScript] public override function GetMonth(time : DateTime) : int;

Description

Gets the month in the specified **System.DateTime** .

Return Value: An integer between 1 and 12 that represents the month in *time* . The **System.DateTime** instance to read.

GetMonthsInYear

[C#] public override int GetMonthsInYear(int year, int era);

[C++] public: int GetMonthsInYear(int year, int era);

[VB] Overrides Public Function GetMonthsInYear(ByVal year As Integer, ByVal era As Integer) As Integer

[JScript] public override function GetMonthsInYear(year : int, era : int) : int; Gets the number of months in the specified year.

Description

Gets the number of months in the year specified by the *year* and *era* parameters.

Return Value: The number of months in the specified year in the specified era. An integer that represents the year. An integer that represents the era.

GetYear

[C#] public override int GetYear(DateTime time);

[C++] public: int GetYear(DateTime time);

[VB] Overrides Public Function GetYear(ByVal time As DateTime) As Integer

[JScript] public override function GetYear(time : DateTime) : int;

Description

Gets the year in the specified **System.DateTime** .

Return Value: An integer between 1 and 9999 that represents the year in *time* . The **System.DateTime** instance to read.

IsLeapDay

[C#] public override bool IsLeapDay(int year, int month, int day, int era);

[C++] public: bool IsLeapDay(int year, int month, int day, int era);

[VB] Overrides Public Function IsLeapDay(ByVal year As Integer, ByVal month As Integer, ByVal day As Integer, ByVal era As Integer) As Boolean

[JScript] public override function IsLeapDay(year : int, month : int, day : int, era : int) : Boolean; Determines whether the specified day is a leap day.

Description

Determines whether the date specified by the *year* , *month* , *day* , and *era* parameters is a leap day.

Return Value: **true** , if the specified day is a leap day; otherwise, **false** .

Leap years in the Japanese calendar correspond to the same leap years in the Gregorian calendar. A common year has 365 days and a leap year has 366 days. An integer that represents the year. An integer that represents the month. An integer that represents the day. An integer that represents the era.

IsLeapMonth

[C#] public override bool IsLeapMonth(int year, int month, int era);

[C++] public: bool IsLeapMonth(int year, int month, int era);

[VB] Overrides Public Function IsLeapMonth(ByVal year As Integer, ByVal month As Integer, ByVal era As Integer) As Boolean

[JScript] public override function IsLeapMonth(year : int, month : int, era : int) : Boolean; Determines whether the specified month is a leap month.

Description

Determines whether the month specified by the *year* , *month* , and *era* parameters is a leap month.

Return Value: This method always returns **false** , unless overridden by a derived class.

Leap years in the Japanese calendar correspond to the same leap years in the Gregorian calendar. A common year has 365 days and a leap year has 366 days. An integer that represents the year. An integer that represents the month. An integer that represents the era.

IsLeapYear

[C#] public override bool IsLeapYear(int year, int era);

[C++] public: bool IsLeapYear(int year, int era);

[VB] Overrides Public Function IsLeapYear(ByVal year As Integer, ByVal era As Integer) As Boolean

[JScript] public override function IsLeapYear(year : int, era : int) : Boolean;

Determines whether the specified year is a leap year.

Description

Determines whether the year specified by the *year* and *era* parameters is a leap year.

Return Value: **true** , if the specified year is a leap year; otherwise, **false** .

Leap years in the Japanese calendar correspond to the same leap years in the Gregorian calendar. A common year has 365 days and a leap year has 366 days. An integer that represents the year. An integer that represents the era.

ToDateTime

[C#] public override DateTime ToDateTime(int year, int month, int day, int hour,


```

1 int minute, int second, int millisecond, int era);
2 [C++] public: DateTime ToDateTime(int year, int month, int day, int hour, int
3 minute, int second, int millisecond, int era);
4 [VB] Overrides Public Function ToDateTime(ByVal year As Integer, ByVal
5 month As Integer, ByVal day As Integer, ByVal hour As Integer, ByVal minute
6 As Integer, ByVal second As Integer, ByVal millisecond As Integer, ByVal era As
7 Integer) As DateTime
8 [JScript] public override function ToDateTime(year : int, month : int, day : int,
9 hour : int, minute : int, second : int, millisecond : int, era : int) : DateTime; Returns
10 a System.DateTime that is set to the specified date.

```

Description

Returns a **System.DateTime** that is set to the specified date and time in the specified era.

Return Value: The **System.DateTime** instance set to the specified date and time in the current era. An integer that represents the year. An integer that represents the month. An integer that represents the day. An integer that represents the hour. An integer that represents the minute. An integer that represents the second. An integer that represents the millisecond. An integer that represents the era.

ToFourDigitYear

```

22 [C#] public override int ToFourDigitYear(int year);
23 [C++] public: int ToFourDigitYear(int year);
24 [VB] Overrides Public Function ToFourDigitYear(ByVal year As Integer) As
25 Integer

```

1 [JScript] public override function ToFourDigitYear(year : int) : int;

2
3 *Description*

4 Converts the specified two-digit year to a four-digit year by using the
5 **System.Globalization.JapaneseCalendar.TwoDigitYearMax** property to
6 determine the appropriate century.

7 *Return Value:* An integer that contains the four-digit representation of *year* .

8 This method implements
9 **System.Globalization.Calendar.ToFourDigitYear(System.Int32)** . A two-digit
10 integer that represents the year to convert.

11 JulianCalendar class (System.Globalization)

12 ToString

13
14
15 *Description*

16 Represents the Julian calendar.

17 In 45 B.C., Julius Caesar ordered a calendar reform, which resulted in the
18 calendar called the Julian calendar. The Julian calendar is the predecessor of the
19 Gregorian calendar.

20 ToString

21
22 [C#] public static readonly int JulianEra;

23 [C++] public: static int JulianEra;

24 [VB] Public Shared ReadOnly JulianEra As Integer

25 [JScript] public static var JulianEra : int;

1
2 *Description*

3 Represents the current era.

4 The **System.Globalization.JulianCalendar** class recognizes only the
5 current era. This field always returns 1.

6 JulianCalendar

7 *Example Syntax:*

8 ToString

9
10 [C#] public JulianCalendar();

11 [C++] public: JulianCalendar();

12 [VB] Public Sub New()

13 [JScript] public function JulianCalendar();

14
15 *Description*

16 Initializes a new instance of the **System.Globalization.JulianCalendar**
17 class.

18 Eras

19 ToString

20
21 [C#] public override int[] Eras {get;}

22 [C++] public: __property virtual int get_Eras();

23 [VB] Overrides Public ReadOnly Property Eras As Integer ()

24 [JScript] public function get Eras() : int[];

Description

Gets the list of eras in the **System.Globalization.JulianCalendar** .

The **System.Globalization.JulianCalendar** class recognizes only the current era. This property always returns an array with only one element.

TwoDigitYearMax

ToString

[C#] public override int TwoDigitYearMax {get; set;}

[C++] public: __property virtual int get_TwoDigitYearMax();public: __property virtual void set_TwoDigitYearMax(int);

[VB] Overrides Public Property TwoDigitYearMax As Integer

[JScript] public function get TwoDigitYearMax() : int;public function set TwoDigitYearMax(int);

Description

Gets or sets the last year of a 100-year range that can be represented by a 2-digit year.

This property allows a 2-digit year to be properly translated to a 4-digit year. For example, if this property is set to 2029, the 100-year range is from 1930 to 2029; therefore, a 2-digit value of 30 is interpreted as 1930, while a 2-digit value of 29 is interpreted as 2029.

AddMonths

[C#] public override DateTime AddMonths(DateTime time, int months);

1 [C++] public: DateTime AddMonths(DateTime time, int months);

2 [VB] Overrides Public Function AddMonths(ByVal time As DateTime, ByVal
3 months As Integer) As DateTime

4 [JScript] public override function AddMonths(time : DateTime, months : int) :
5 DateTime;

6 7 *Description*

8 Returns a **System.DateTime** that is the specified number of months away
9 from the specified **System.DateTime** .

10 *Return Value:* The **System.DateTime** that results from adding the specified
11 number of months to the specified **System.DateTime** .

12 The year part of the resulting **System.DateTime** is affected if the resulting
13 month is beyond the last month of the current year. The day part of the resulting
14 **System.DateTime** is also affected if the resulting day is not a valid day in the
15 resulting month of the resulting year; it is changed to the last valid day in the
16 resulting month of the resulting year. The time-of-day part of the resulting
17 **System.DateTime** remains the same as the specified **System.DateTime** . The
18 **System.DateTime** instance to add. The number of months to add.

19 *AddYears*

20
21 [C#] public override DateTime AddYears(DateTime time, int years);

22 [C++] public: DateTime AddYears(DateTime time, int years);

23 [VB] Overrides Public Function AddYears(ByVal time As DateTime, ByVal years
24 As Integer) As DateTime

25 [JScript] public override function AddYears(time : DateTime, years : int) :

DateTime;

Description

Returns a **System.DateTime** that is the specified number of years away from the specified **System.DateTime**.

Return Value: The **System.DateTime** that results from adding the specified number of years to the specified **System.DateTime**.

The day part of the resulting **System.DateTime** is affected if the resulting day is not a valid day in the resulting month of the resulting year; it is changed to the last valid day in the resulting month of the resulting year. The time-of-day part of the resulting **System.DateTime** remains the same as the specified **System.DateTime**. The **System.DateTime** instance to add. The number of years to add.

GetDayOfMonth

[C#] public override int GetDayOfMonth(DateTime time);

[C++] public: int GetDayOfMonth(DateTime time);

[VB] Overrides Public Function GetDayOfMonth(ByVal time As DateTime) As Integer

[JScript] public override function GetDayOfMonth(time : DateTime) : int;

Description

Gets the day of the month in the specified **System.DateTime**.

Return Value: An integer from 1 to 31 that represents the day of the month in *time*. The **System.DateTime** instance to read.

GetDayOfWeek

[C#] public override DayOfWeek GetDayOfWeek(DateTime time);
[C++] public: DayOfWeek GetDayOfWeek(DateTime time);
[VB] Overrides Public Function GetDayOfWeek(ByVal time As DateTime) As
DayOfWeek
[JScript] public override function GetDayOfWeek(time : DateTime) :
DayOfWeek;

Description

Gets the day of the week in the specified **System.DateTime** .

Return Value: A **System.DayOfWeek** value that represents the day of the week in
time .

The **System.DayOfWeek** values are Sunday, Monday, Tuesday,
Wednesday, Thursday, Friday, and Saturday. The **System.DateTime** instance to
read.

GetDayOfYear

[C#] public override int GetDayOfYear(DateTime time);
[C++] public: int GetDayOfYear(DateTime time);
[VB] Overrides Public Function GetDayOfYear(ByVal time As DateTime) As
Integer
[JScript] public override function GetDayOfYear(time : DateTime) : int;

Description

Gets the day of the year in the specified **System.DateTime** .

Return Value: An integer from 1 to 366 that represents the day of the year in *time* .

The **System.DateTime** instance to read.

GetDaysInMonth

[C#] public override int GetDaysInMonth(int year, int month, int era);

[C++] public: int GetDaysInMonth(int year, int month, int era);

[VB] Overrides Public Function GetDaysInMonth(ByVal year As Integer, ByVal month As Integer, ByVal era As Integer) As Integer

[JScript] public override function GetDaysInMonth(year : int, month : int, era : int) : int; Gets the number of days in the specified month.

Description

Gets the number of days in the month specified by the *year* , *month* , and *era* parameters.

Return Value: The number of days in the specified month in the specified year in the specified era.

For example, this method might return 28 or 29 for February (*month* = 2), depending on whether *year* is a leap year. An integer that represents the year. An integer that represents the month. An integer that represents the era.

GetDaysInYear

[C#] public override int GetDaysInYear(int year, int era);

[C++] public: int GetDaysInYear(int year, int era);

[VB] Overrides Public Function GetDaysInYear(ByVal year As Integer, ByVal

era As Integer) As Integer

[JScript] public override function GetDaysInYear(year : int, era : int) : int; Gets the number of days in the specified year.

Description

Gets the number of days in the year specified by the *year* and *era* parameters.

Return Value: The number of days in the specified year in the specified era.

For example, this method might return 365 or 366, depending on whether *year* is a leap year. An integer that represents the year. An integer that represents the era.

GetEra

[C#] public override int GetEra(DateTime time);

[C++] public: int GetEra(DateTime time);

[VB] Overrides Public Function GetEra(ByVal time As DateTime) As Integer

[JScript] public override function GetEra(time : DateTime) : int;

Description

Gets the era in the specified **System.DateTime** .

Return Value: An integer that represents the era in *time* .

The **System.Globalization.JulianCalendar** class recognizes only the current era. The **System.DateTime** instance to read.

GetMonth

1
2 [C#] public override int GetMonth(DateTime time);

3 [C++] public: int GetMonth(DateTime time);

4 [VB] Overrides Public Function GetMonth(ByVal time As DateTime) As Integer

5 [JScript] public override function GetMonth(time : DateTime) : int;

6
7 *Description*

8 Gets the month in the specified **System.DateTime** .

9 *Return Value:* An integer between 1 and 12 that represents the month in *time* . The

10 **System.DateTime** instance to read.

11 **GetMonthsInYear**

12
13 [C#] public override int GetMonthsInYear(int year, int era);

14 [C++] public: int GetMonthsInYear(int year, int era);

15 [VB] Overrides Public Function GetMonthsInYear(ByVal year As Integer, ByVal
16 era As Integer) As Integer

17 [JScript] public override function GetMonthsInYear(year : int, era : int) : int; Gets
18 the number of months in the specified year.

19
20 *Description*

21 Gets the number of months in the year specified by the *year* and *era*
22 parameters.

23 *Return Value:* The number of months in the specified year in the specified era. An
24 integer that represents the year. An integer that represents the era.

25 **GetYear**

[C#] public override int GetYear(DateTime time);

[C++] public: int GetYear(DateTime time);

[VB] Overrides Public Function GetYear(ByVal time As DateTime) As Integer

[JScript] public override function GetYear(time : DateTime) : int;

Description

Gets the year in the specified **System.DateTime** .

Return Value: An integer between 1 and 9999 that represents the year in *time* . The **System.DateTime** instance to read.

IsLeapDay

[C#] public override bool IsLeapDay(int year, int month, int day, int era);

[C++] public: bool IsLeapDay(int year, int month, int day, int era);

[VB] Overrides Public Function IsLeapDay(ByVal year As Integer, ByVal month As Integer, ByVal day As Integer, ByVal era As Integer) As Boolean

[JScript] public override function IsLeapDay(year : int, month : int, day : int, era : int) : Boolean; Determines whether the specified day is a leap day.

Description

Determines whether the date specified by the *year* , *month* , *day* , and *era* parameters is a leap day.

Return Value: **true** if the specified day is a leap day; otherwise, **false** .

Unlike the Gregorian calendar, the Julian calendar defines a leap year as a year that is evenly divisible by four with no exceptions; therefore, the calendar is

inaccurate by one day every 128 years. For example, the year 1999 was not a leap year, but the year 2000 was. A common year has 365 days and a leap year has 366 days. An integer that represents the year. An integer that represents the month. An integer that represents the day. An integer that represents the era.

IsLeapMonth

[C#] public override bool IsLeapMonth(int year, int month, int era);

[C++] public: bool IsLeapMonth(int year, int month, int era);

[VB] Overrides Public Function IsLeapMonth(ByVal year As Integer, ByVal month As Integer, ByVal era As Integer) As Boolean

[JScript] public override function IsLeapMonth(year : int, month : int, era : int) : Boolean; Determines whether the specified month is a leap month.

Description

Determines whether the month specified by the *year*, *month*, and *era* parameters is a leap month.

Return Value: This method always returns **false**, unless overridden by a derived class.

Unlike the Gregorian calendar, the Julian calendar defines a leap year as a year that is evenly divisible by four with no exceptions; therefore, the calendar is inaccurate by one day every 128 years. For example, the year 1999 was not a leap year, but the year 2000 was. A common year has 365 days and a leap year has 366 days. An integer that represents the year. An integer that represents the month. An integer that represents the era.

IsLeapYear

1
2 [C#] public override bool IsLeapYear(int year, int era);

3 [C++] public: bool IsLeapYear(int year, int era);

4 [VB] Overrides Public Function IsLeapYear(ByVal year As Integer, ByVal era As
5 Integer) As Boolean

6 [JScript] public override function IsLeapYear(year : int, era : int) : Boolean;

7 Determines whether the specified year is a leap year.

8
9 *Description*

10 Determines whether the year specified by the *year* and *era* parameters is a
11 leap year.

12 *Return Value:* **true** if the specified year is a leap year; otherwise, **false** .

13 Unlike the Gregorian calendar, the Julian calendar defines a leap year as a
14 year that is evenly divisible by four with no exceptions; therefore, the calendar is
15 inaccurate by one day every 128 years. For example, the year 1999 was not a leap
16 year, but the year 2000 was. A common year has 365 days and a leap year has 366
17 days. An integer that represents the year. An integer that represents the era.

18 *ToDateTime*

19
20 [C#] public override DateTime ToDateTime(int year, int month, int day, int hour,
21 int minute, int second, int millisecond, int era);

22 [C++] public: DateTime ToDateTime(int year, int month, int day, int hour, int
23 minute, int second, int millisecond, int era);

24 [VB] Overrides Public Function ToDateTime(ByVal year As Integer, ByVal
25 month As Integer, ByVal day As Integer, ByVal hour As Integer, ByVal minute

As Integer, ByVal second As Integer, ByVal millisecond As Integer, ByVal era As Integer) As DateTime

[JScript] public override function ToDateTime(year : int, month : int, day : int, hour : int, minute : int, second : int, millisecond : int, era : int) : DateTime; Returns a **System.DateTime** that is set to the specified date.

Description

Returns a **System.DateTime** that is set to the specified date and time in the specified era.

Return Value: The **System.DateTime** instance set to the specified date and time in the current era. An integer that represents the year. An integer that represents the month. An integer that represents the day. An integer that represents the hour. An integer that represents the minute. An integer that represents the second. An integer that represents the millisecond. An integer that represents the era.

ToFourDigitYear

[C#] public override int ToFourDigitYear(int year);

[C++] public: int ToFourDigitYear(int year);

[VB] Overrides Public Function ToFourDigitYear(ByVal year As Integer) As Integer

[JScript] public override function ToFourDigitYear(year : int) : int;

Description

Converts the specified two-digit year to a four-digit year by using the **System.Globalization.JulianCalendar.TwoDigitYearMax** property to determine

the appropriate century.

Return Value: An integer that contains the four-digit representation of *year* .

System.Globalization.JulianCalendar.TwoDigitYearMax is the last year in the 100-year range that can be represented by a two-digit year. The century is determined by finding the sole occurrence of the two-digit *year* within that 100-year range. For example, if

System.Globalization.JulianCalendar.TwoDigitYearMax is set to 2029, the 100-year range is from 1930 to 2029; therefore, a 2-digit value of 30 is interpreted as 1930, while a 2-digit value of 29 is interpreted as 2029. A two-digit integer that represents the year to convert.

KoreanCalendar class (System.Globalization)

ToString

Description

Represents the Korean calendar.

The Korean calendar works exactly like the Gregorian calendar, except that the year and era are different.

ToString

[C#] public const int KoreanEra;

[C++] public: const int KoreanEra;

[VB] Public Const KoreanEra As Integer

[JScript] public var KoreanEra : int;

1
2 *Description*

3 Represents the current era.

4 The **System.Globalization.KoreanCalendar** class recognizes only the
5 current era. This field always returns 1.

6 KoreanCalendar

7 *Example Syntax:*

8 ToString

9
10 [C#] public KoreanCalendar();

11 [C++] public: KoreanCalendar();

12 [VB] Public Sub New()

13 [JScript] public function KoreanCalendar();

14
15 *Description*

16 Initializes a new instance of the **System.Globalization.KoreanCalendar**
17 class.

18 Eras

19 ToString

20
21 [C#] public override int[] Eras {get;}

22 [C++] public: __property virtual int get_Eras();

23 [VB] Overrides Public ReadOnly Property Eras As Integer ()

24 [JScript] public function get Eras() : int[];

Description

Gets the list of eras in the **System.Globalization.KoreanCalendar** .

The **System.Globalization.KoreanCalendar** class recognizes only the current era. This property always returns an array with only one element.

TwoDigitYearMax

ToString

```
[C#] public override int TwoDigitYearMax {get; set;}
```

```
[C++] public: __property virtual int get_TwoDigitYearMax();public: __property  
virtual void set_TwoDigitYearMax(int);
```

```
[VB] Overrides Public Property TwoDigitYearMax As Integer
```

```
[JScript] public function get TwoDigitYearMax() : int;public function set  
TwoDigitYearMax(int);
```

Description

Gets or sets the last year of a 100-year range that can be represented by a 2-digit year.

This property allows a 2-digit year to be properly translated to a 4-digit year. For example, in the Gregorian calendar, if this property is set to 2029, the 100-year range is from 1930 to 2029; therefore, a 2-digit value of 30 is interpreted as 1930, while a 2-digit value of 29 is interpreted as 2029.

AddMonths

```
[C#] public override DateTime AddMonths(DateTime time, int months);
```

1 [C++] public: DateTime AddMonths(DateTime time, int months);

2 [VB] Overrides Public Function AddMonths(ByVal time As DateTime, ByVal
3 months As Integer) As DateTime

4 [JScript] public override function AddMonths(time : DateTime, months : int) :
5 DateTime;

6 7 *Description*

8 Returns a **System.DateTime** that is the specified number of months away
9 from the specified **System.DateTime** .

10 *Return Value:* The **System.DateTime** that results from adding the specified
11 number of months to the specified **System.DateTime** .

12 The year part of the resulting **System.DateTime** is affected if the resulting
13 month is beyond the last month of the current year. The day part of the resulting
14 **System.DateTime** is also affected if the resulting day is not a valid day in the
15 resulting month of the resulting year; it is changed to the last valid day in the
16 resulting month of the resulting year. The time-of-day part of the resulting
17 **System.DateTime** remains the same as the specified **System.DateTime** . The
18 **System.DateTime** instance to add. The number of months to add.

19 *AddYears*

20
21 [C#] public override DateTime AddYears(DateTime time, int years);

22 [C++] public: DateTime AddYears(DateTime time, int years);

23 [VB] Overrides Public Function AddYears(ByVal time As DateTime, ByVal years
24 As Integer) As DateTime

25 [JScript] public override function AddYears(time : DateTime, years : int) :

DateTime;

Description

Returns a **System.DateTime** that is the specified number of years away from the specified **System.DateTime**.

Return Value: The **System.DateTime** that results from adding the specified number of years to the specified **System.DateTime**.

The day part of the resulting **System.DateTime** is affected if the resulting day is not a valid day in the resulting month of the resulting year; it is changed to the last valid day in the resulting month of the resulting year. The time-of-day part of the resulting **System.DateTime** remains the same as the specified **System.DateTime**. The **System.DateTime** instance to add. The number of years to add.

GetDayOfMonth

[C#] public override int GetDayOfMonth(DateTime time);

[C++] public: int GetDayOfMonth(DateTime time);

[VB] Overrides Public Function GetDayOfMonth(ByVal time As DateTime) As

Integer

[JScript] public override function GetDayOfMonth(time : DateTime) : int;

Description

Gets the day of the month in the specified **System.DateTime**.

Return Value: An integer from 1 to 31 that represents the day of the month in *time*. The **System.DateTime** instance to read.

GetDayOfWeek

[C#] public override DayOfWeek GetDayOfWeek(DateTime time);
[C++] public: DayOfWeek GetDayOfWeek(DateTime time);
[VB] Overrides Public Function GetDayOfWeek(ByVal time As DateTime) As
DayOfWeek
[JScript] public override function GetDayOfWeek(time : DateTime) :
DayOfWeek;

Description

Gets the day of the week in the specified **System.DateTime** .

Return Value: A **System.DayOfWeek** value that represents the day of the week in
time .

The **System.DayOfWeek** values are Sunday, Monday, Tuesday,
Wednesday, Thursday, Friday, and Saturday. The **System.DateTime** instance to
read.

GetDayOfYear

[C#] public override int GetDayOfYear(DateTime time);
[C++] public: int GetDayOfYear(DateTime time);
[VB] Overrides Public Function GetDayOfYear(ByVal time As DateTime) As
Integer
[JScript] public override function GetDayOfYear(time : DateTime) : int;

Description

Gets the day of the year in the specified **System.DateTime** .

Return Value: An integer from 1 to 366 that represents the day of the year in *time* .

The **System.DateTime** instance to read.

GetDaysInMonth

[C#] public override int GetDaysInMonth(int year, int month, int era);

[C++] public: int GetDaysInMonth(int year, int month, int era);

[VB] Overrides Public Function GetDaysInMonth(ByVal year As Integer, ByVal month As Integer, ByVal era As Integer) As Integer

[JScript] public override function GetDaysInMonth(year : int, month : int, era : int) : int; Gets the number of days in the specified month.

Description

Gets the number of days in the month specified by the *year* , *month* , and *era* parameters.

Return Value: The number of days in the specified month in the specified year in the specified era.

For example, this method might return 28 or 29 for February, *month* = 2), depending on whether *year* is a leap year. An integer that represents the year. An integer that represents the month. An integer that represents the era.

GetDaysInYear

[C#] public override int GetDaysInYear(int year, int era);

[C++] public: int GetDaysInYear(int year, int era);

[VB] Overrides Public Function GetDaysInYear(ByVal year As Integer, ByVal

era As Integer) As Integer

[JScript] public override function GetDaysInYear(year : int, era : int) : int; Gets the number of days in the specified year.

Description

Gets the number of days in the year specified by the *year* and *era* parameters.

Return Value: The number of days in the specified year in the specified era.

For example, this method might return 365 or 366, depending on whether *year* is a leap year. An integer that represents the year. An integer that represents the era.

GetEra

[C#] public override int GetEra(DateTime time);

[C++] public: int GetEra(DateTime time);

[VB] Overrides Public Function GetEra(ByVal time As DateTime) As Integer

[JScript] public override function GetEra(time : DateTime) : int;

Description

Gets the era in the specified **System.DateTime** .

Return Value: An integer that represents the era in *time* .

The **System.Globalization.KoreanCalendar** class recognizes only the current era. The **System.DateTime** instance to read.

GetMonth

1
2 [C#] public override int GetMonth(DateTime time);

3 [C++] public: int GetMonth(DateTime time);

4 [VB] Overrides Public Function GetMonth(ByVal time As DateTime) As Integer

5 [JScript] public override function GetMonth(time : DateTime) : int;

6
7 *Description*

8 Gets the month in the specified **System.DateTime** .

9 *Return Value:* An integer between 1 and 12 that represents the month in *time* . The
10 **System.DateTime** instance to read.

11 **GetMonthsInYear**

12
13 [C#] public override int GetMonthsInYear(int year, int era);

14 [C++] public: int GetMonthsInYear(int year, int era);

15 [VB] Overrides Public Function GetMonthsInYear(ByVal year As Integer, ByVal
16 era As Integer) As Integer

17 [JScript] public override function GetMonthsInYear(year : int, era : int) : int; Gets
18 the number of months in the specified year.

19
20 *Description*

21 Gets the number of months in the year specified by the *year* and *era*
22 parameters.

23 *Return Value:* The number of months in the specified year in the specified era. An
24 integer that represents the year. An integer that represents the era.

25 **GetYear**

1
2 [C#] public override int GetYear(DateTime time);

3 [C++] public: int GetYear(DateTime time);

4 [VB] Overrides Public Function GetYear(ByVal time As DateTime) As Integer

5 [JScript] public override function GetYear(time : DateTime) : int;

6
7 *Description*

8 Gets the year in the specified **System.DateTime** .

9 *Return Value:* An integer between 1 and 9999 that represents the year in *time* . The
10 **System.DateTime** instance to read.

11 **IsLeapDay**

12
13 [C#] public override bool IsLeapDay(int year, int month, int day, int era);

14 [C++] public: bool IsLeapDay(int year, int month, int day, int era);

15 [VB] Overrides Public Function IsLeapDay(ByVal year As Integer, ByVal month
16 As Integer, ByVal day As Integer, ByVal era As Integer) As Boolean

17 [JScript] public override function IsLeapDay(year : int, month : int, day : int, era :
18 int) : Boolean; Determines whether the specified day is a leap day.

19
20 *Description*

21 Determines whether the date specified by the *year* , *month* , *day* , and *era*
22 parameters is a leap day.

23 *Return Value:* **true** if the specified day is a leap day; otherwise, **false** .

24 Leap years in the Korean calendar correspond to the same leap years in the
25 Gregorian calendar. A common year has 365 days and a leap year has 366 days.

An integer that represents the year. An integer that represents the month. An integer that represents the day. An integer that represents the era.

IsLeapMonth

[C#] public override bool IsLeapMonth(int year, int month, int era);
 [C++] public: bool IsLeapMonth(int year, int month, int era);
 [VB] Overrides Public Function IsLeapMonth(ByVal year As Integer, ByVal month As Integer, ByVal era As Integer) As Boolean
 [JScript] public override function IsLeapMonth(year : int, month : int, era : int) : Boolean; Determines whether the specified month is a leap month.

Description

Determines whether the month specified by the *year* , *month* , and *era* parameters is a leap month.

Return Value: This method always returns **false** , unless overridden by a derived class.

Leap years in the Korean calendar correspond to the same leap years in the Gregorian calendar. A common year has 365 days and a leap year has 366 days.

An integer that represents the year. An integer that represents the month. An integer that represents the era.

IsLeapYear

[C#] public override bool IsLeapYear(int year, int era);
 [C++] public: bool IsLeapYear(int year, int era);
 [VB] Overrides Public Function IsLeapYear(ByVal year As Integer, ByVal era As

Integer) As Boolean

[JScript] public override function IsLeapYear(year : int, era : int) : Boolean;

Determines whether the specified year is a leap year.

Description

Determines whether the year specified by the *year* and *era* parameters is a leap year.

Return Value: **true** if the specified year is a leap year; otherwise, **false** .

Leap years in the Korean calendar correspond to the same leap years in the Gregorian calendar. A common year has 365 days and a leap year has 366 days.

An integer that represents the year. An integer that represents the era.

ToDateTime

[C#] public override DateTime ToDateTime(int year, int month, int day, int hour, int minute, int second, int millisecond, int era);

[C++] public: DateTime ToDateTime(int year, int month, int day, int hour, int minute, int second, int millisecond, int era);

[VB] Overrides Public Function ToDateTime(ByVal year As Integer, ByVal month As Integer, ByVal day As Integer, ByVal hour As Integer, ByVal minute As Integer, ByVal second As Integer, ByVal millisecond As Integer, ByVal era As Integer) As DateTime

[JScript] public override function ToDateTime(year : int, month : int, day : int, hour : int, minute : int, second : int, millisecond : int, era : int) : DateTime; Returns a **System.DateTime** that is set to the specified date.

Description

Returns a **System.DateTime** that is set to the specified date and time in the specified era.

Return Value: The **System.DateTime** instance set to the specified date and time in the current era. An integer that represents the year. An integer that represents the month. An integer that represents the day. An integer that represents the hour. An integer that represents the minute. An integer that represents the second. An integer that represents the millisecond. An integer that represents the era.

ToFourDigitYear

[C#] public override int ToFourDigitYear(int year);

[C++] public: int ToFourDigitYear(int year);

[VB] Overrides Public Function ToFourDigitYear(ByVal year As Integer) As Integer

[JScript] public override function ToFourDigitYear(year : int) : int;

Description

Converts the specified two-digit year to a four-digit year by using the **System.Globalization.KoreanCalendar.TwoDigitYearMax** property to determine the appropriate century.

Return Value: An integer that contains the four-digit representation of *year* .

System.Globalization.KoreanCalendar.TwoDigitYearMax is the last year in the 100-year range that can be represented by a two-digit year. The century is determined by finding the sole occurrence of the two-digit *year* within that 100-

year range. For example, if

System.Globalization.KoreanCalendar.TwoDigitYearMax is set to 2029, the 100-year range is from 1930 to 2029; therefore, a 2-digit value of 30 is interpreted as 1930, while a 2-digit value of 29 is interpreted as 2029. A two-digit integer that represents the year to convert.

NumberFormatInfo class (System.Globalization)

ToString

Description

Defines how numeric values are formatted and displayed, depending on the culture.

This class contains information, such as currency, decimal separators, and other numeric symbols. Numeric values are formatted using standard or custom patterns stored in the properties of a **System.Globalization.NumberFormatInfo** instance. To modify how a value is displayed, the

System.Globalization.NumberFormatInfo instance must be writable so custom patterns can be saved in its properties.

NumberFormatInfo

Example Syntax:

ToString

[C#] public NumberFormatInfo();

[C++] public: NumberFormatInfo();

[VB] Public Sub New()

1 [JScript] public function NumberFormatInfo();

3 *Description*

4 Initializes a new writable instance of the
5 **System.Globalization.NumberFormatInfo** class that is culture-independent
6 (invariant).

7 The properties of the new instance can be modified if you want user-
8 defined formatting.

9 CurrencyDecimalDigits

10 ToString

12 [C#] public int CurrencyDecimalDigits {get; set;}

13 [C++] public: __property int get_CurrencyDecimalDigits();public: __property
14 void set_CurrencyDecimalDigits(int);

15 [VB] Public Property CurrencyDecimalDigits As Integer

16 [JScript] public function get CurrencyDecimalDigits() : int;public function set
17 CurrencyDecimalDigits(int);

19 *Description*

20 Indicates the number of decimal places to use in currency values.

21 CurrencyDecimalSeparator

22 ToString

24 [C#] public string CurrencyDecimalSeparator {get; set;}

25 [C++] public: __property String* get_CurrencyDecimalSeparator();public:

1 __property void set_CurrencyDecimalSeparator(String*);

2 [VB] Public Property CurrencyDecimalSeparator As String

3 [JScript] public function get CurrencyDecimalSeparator() : String;public function

4 set CurrencyDecimalSeparator(String);

5
6 *Description*

7 Indicates the **System.String** to use as the decimal separator in currency
8 values.

9 CurrencyGroupSeparator

10 ToString

11
12 [C#] public string CurrencyGroupSeparator {get; set;}

13 [C++] public: __property String* get_CurrencyGroupSeparator();public:

14 __property void set_CurrencyGroupSeparator(String*);

15 [VB] Public Property CurrencyGroupSeparator As String

16 [JScript] public function get CurrencyGroupSeparator() : String;public function set

17 CurrencyGroupSeparator(String);

18
19 *Description*

20 Indicates the **System.String** that separates groups of digits to the left of the
21 decimal in currency values.

22 CurrencyGroupSizes

23 ToString

24
25 [C#] public int[] CurrencyGroupSizes {get; set;}

```

1 [C++] public: __property int get_CurrencyGroupSizes();public: __property void
2 set_CurrencyGroupSizes(int __gc[]);
3 [VB] Public Property CurrencyGroupSizes As Integer ()
4 [JScript] public function get CurrencyGroupSizes() : int[];public function set
5 CurrencyGroupSizes(int[]);

```

Description

Indicates the number of digits in each group to the left of the decimal in currency values.

Every element in the one-dimensional array must be an integer from 1 through 9. The last element can be 0.

CurrencyNegativePattern

ToString

```

15 [C#] public int CurrencyNegativePattern {get; set;}
16 [C++] public: __property int get_CurrencyNegativePattern();public: __property
17 void set_CurrencyNegativePattern(int);
18 [VB] Public Property CurrencyNegativePattern As Integer
19 [JScript] public function get CurrencyNegativePattern() : int;public function set
20 CurrencyNegativePattern(int);

```

Description

Indicates the format pattern for negative currency values.

This property can have one of the values in the following table. The symbol "\$" is the **System.Globalization.NumberFormatInfo.CurrencySymbol**, the

1 symbol "-" is the **System.Globalization.NumberFormatInfo.NegativeSign** , and
2 *n* is a number.

3 CurrencyPositivePattern

4 ToString

6 [C#] public int CurrencyPositivePattern {get; set;}

7 [C++] public: __property int get_CurrencyPositivePattern();public: __property
8 void set_CurrencyPositivePattern(int);

9 [VB] Public Property CurrencyPositivePattern As Integer

10 [JScript] public function get CurrencyPositivePattern() : int;public function set
11 CurrencyPositivePattern(int);

13 *Description*

14 Indicates the format pattern for positive currency values.

15 This property can have one of the values in the following table. The symbol
16 "\$" is the **System.Globalization.NumberFormatInfo.CurrencySymbol** and *n* is
17 a number.

18 CurrencySymbol

19 ToString

21 [C#] public string CurrencySymbol {get; set;}

22 [C++] public: __property String* get_CurrencySymbol();public: __property void
23 set_CurrencySymbol(String*);

24 [VB] Public Property CurrencySymbol As String

25 [JScript] public function get CurrencySymbol() : String;public function set

1 CurrencySymbol(String);

3 *Description*

4 Indicates the **System.String** to use as the currency symbol.

5 CurrentInfo

6 ToString

8 [C#] public static NumberFormatInfo CurrentInfo {get;}

9 [C++] public: __property static NumberFormatInfo* get_CurrentInfo();

10 [VB] Public Shared ReadOnly Property CurrentInfo As NumberFormatInfo

11 [JScript] public static function get CurrentInfo() : NumberFormatInfo;

13 *Description*

14 Gets a read-only **System.Globalization.NumberFormatInfo** instance that
15 formats values based on the current culture.

16 InvariantInfo

17 ToString

19 [C#] public static NumberFormatInfo InvariantInfo {get;}

20 [C++] public: __property static NumberFormatInfo* get_InvariantInfo();

21 [VB] Public Shared ReadOnly Property InvariantInfo As NumberFormatInfo

22 [JScript] public static function get InvariantInfo() : NumberFormatInfo;

24 *Description*

1 Gets the default read-only **System.Globalization.NumberFormatInfo**
2 instance that is culture-independent (invariant).

3 This property does not change, regardless of the current culture.

4 **IsReadOnly**

5 **ToString**

6
7 [C#] public bool IsReadOnly {get;}

8 [C++] public: __property bool get_IsReadOnly();

9 [VB] Public ReadOnly Property IsReadOnly As Boolean

10 [JScript] public function get IsReadOnly() : Boolean;

11
12 *Description*

13 Gets a value indicating whether the
14 **System.Globalization.NumberFormatInfo** is read-only.

15 **NaNSymbol**

16 **ToString**

17
18 [C#] public string NaNSymbol {get; set;}

19 [C++] public: __property String* get_NaNSymbol();public: __property void
20 set_NaNSymbol(String*);

21 [VB] Public Property NaNSymbol As String

22 [JScript] public function get NaNSymbol() : String;public function set
23 NaNSymbol(String);

24
25 *Description*

Indicates the **System.String** that represents the IEEE NaN (not a number) value.

NegativeInfinitySymbol

ToString

[C#] public string NegativeInfinitySymbol {get; set;}

[C++] public: __property String* get_NegativeInfinitySymbol();public:

__property void set_NegativeInfinitySymbol(String*);

[VB] Public Property NegativeInfinitySymbol As String

[JScript] public function get NegativeInfinitySymbol() : String;public function set

NegativeInfinitySymbol(String);

Description

Indicates the **System.String** that represents negative infinity.

NegativeSign

ToString

[C#] public string NegativeSign {get; set;}

[C++] public: __property String* get_NegativeSign();public: __property void

set_NegativeSign(String*);

[VB] Public Property NegativeSign As String

[JScript] public function get NegativeSign() : String;public function set

NegativeSign(String);

Description

Indicates the **System.String** that denotes that the associated number is negative.

NumberDecimalDigits

ToString

[C#] public int NumberDecimalDigits {get; set;}

[C++] public: __property int get_NumberDecimalDigits();public: __property void set_NumberDecimalDigits(int);

[VB] Public Property NumberDecimalDigits As Integer

[JScript] public function get NumberDecimalDigits() : int;public function set NumberDecimalDigits(int);

Description

Indicates the number of decimal places to use in numeric values.

NumberDecimalSeparator

ToString

[C#] public string NumberDecimalSeparator {get; set;}

[C++] public: __property String* get_NumberDecimalSeparator();public: __property void set_NumberDecimalSeparator(String*);

[VB] Public Property NumberDecimalSeparator As String

[JScript] public function get NumberDecimalSeparator() : String;public function set NumberDecimalSeparator(String);

Description

Indicates the **System.String** to use as the decimal separator in numeric values.

NumberGroupSeparator

ToString

[C#] public string NumberGroupSeparator {get; set;}

[C++] public: __property String* get_NumberGroupSeparator();public:

__property void set_NumberGroupSeparator(String*);

[VB] Public Property NumberGroupSeparator As String

[JScript] public function get NumberGroupSeparator() : String;public function set

NumberGroupSeparator(String);

Description

Indicates the **System.String** that separates groups of digits to the left of the decimal in numeric values.

NumberGroupSizes

ToString

[C#] public int[] NumberGroupSizes {get; set;}

[C++] public: __property int get_NumberGroupSizes();public: __property void

set_NumberGroupSizes(int __gc[]);

[VB] Public Property NumberGroupSizes As Integer ()

[JScript] public function get NumberGroupSizes() : int[];public function set

NumberGroupSizes(int[]);

1
2 *Description*

3 Indicates the number of digits in each group to the left of the decimal in
4 numeric values.

5 Every element in the one-dimensional array must be an integer from 1
6 through 9. The last element can be 0.

7 NumberNegativePattern

8 ToString

9
10 [C#] public int NumberNegativePattern {get; set;}

11 [C++] public: __property int get _NumberNegativePattern();public: __property
12 void set _NumberNegativePattern(int);

13 [VB] Public Property NumberNegativePattern As Integer

14 [JScript] public function get NumberNegativePattern() : int;public function set
15 NumberNegativePattern(int);

16
17 *Description*

18 Indicates the format pattern for negative numeric values.

19 This property can have one of the values in the following table. The symbol
20 "-" is the **System.Globalization.NumberFormatInfo.NegativeSign** and *n* is a
21 number.

22 PercentDecimalDigits

23 ToString

24
25 [C#] public int PercentDecimalDigits {get; set;}

```

1 [C++] public: __property int get_PercentDecimalDigits();public: __property void
2 set_PercentDecimalDigits(int);
3 [VB] Public Property PercentDecimalDigits As Integer
4 [JScript] public function get PercentDecimalDigits() : int;public function set
5 PercentDecimalDigits(int);

```

Description

Indicates the number of decimal places to use in percent values.

PercentDecimalSeparator

ToString

```

12 [C#] public string PercentDecimalSeparator {get; set;}

```

```

13 [C++] public: __property String* get_PercentDecimalSeparator();public:
14 __property void set_PercentDecimalSeparator(String*);

```

```

15 [VB] Public Property PercentDecimalSeparator As String

```

```

16 [JScript] public function get PercentDecimalSeparator() : String;public function
17 set PercentDecimalSeparator(String);

```

Description

Indicates the **System.String** to use as the decimal separator in percent values.

PercentGroupSeparator

ToString

```

25 [C#] public string PercentGroupSeparator {get; set;}

```

```

1 [C++] public: __property String* get_PercentGroupSeparator();public: __property
2 void set_PercentGroupSeparator(String*);
3 [VB] Public Property PercentGroupSeparator As String
4 [JScript] public function get PercentGroupSeparator() : String;public function set
5 PercentGroupSeparator(String);
6

```

Description

Indicates the **System.String** that separates groups of digits to the left of the decimal in percent values.

PercentGroupSizes

ToString

```

13 [C#] public int[] PercentGroupSizes {get; set;}

```

```

14 [C++] public: __property int get_PercentGroupSizes();public: __property void
15 set_PercentGroupSizes(int __gc[]);

```

```

16 [VB] Public Property PercentGroupSizes As Integer ()

```

```

17 [JScript] public function get PercentGroupSizes() : int[];public function set
18 PercentGroupSizes(int[]);

```

Description

Indicates the number of digits in each group to the left of the decimal in percent values.

Every element in the one-dimensional array must be an integer from 1 through 9. The last element can be 0.

PercentNegativePattern

ToString

[C#] public int PercentNegativePattern {get; set;}

[C++] public: __property int get_PercentNegativePattern();public: __property void set_PercentNegativePattern(int);

[VB] Public Property PercentNegativePattern As Integer

[JScript] public function get PercentNegativePattern() : int;public function set PercentNegativePattern(int);

Description

Indicates the format pattern for negative percent values.

This property can have one of the values in the following table. The symbol "%" is the **System.Globalization.NumberFormatInfo.PercentSymbol**, the symbol "-" is the **System.Globalization.NumberFormatInfo.NegativeSign**, and *n* is a number.

PercentPositivePattern

ToString

[C#] public int PercentPositivePattern {get; set;}

[C++] public: __property int get_PercentPositivePattern();public: __property void set_PercentPositivePattern(int);

[VB] Public Property PercentPositivePattern As Integer

[JScript] public function get PercentPositivePattern() : int;public function set PercentPositivePattern(int);

1
2 *Description*

3 Indicates the format pattern for positive percent values.

4 This property can have one of the values in the following table. The symbol
5 "%" is the **System.Globalization.NumberFormatInfo.PercentSymbol** and *n* is a
6 number.

7 PercentSymbol

8 ToString

9
10 [C#] public string PercentSymbol {get; set;}

11 [C++] public: __property String* get_PercentSymbol();public: __property void
12 set_PercentSymbol(String*);

13 [VB] Public Property PercentSymbol As String

14 [JScript] public function get PercentSymbol() : String;public function set
15 PercentSymbol(String);

16
17 *Description*

18 Indicates the **System.String** to use as the percent symbol.

19 PerMilleSymbol

20 ToString

21
22 [C#] public string PerMilleSymbol {get; set;}

23 [C++] public: __property String* get_PerMilleSymbol();public: __property void
24 set_PerMilleSymbol(String*);

25 [VB] Public Property PerMilleSymbol As String

1 [JScript] public function get PerMilleSymbol() : String;public function set
2 PerMilleSymbol(String);
3

4 *Description*

5 Indicates the **System.String** to use as the per mille symbol.

6 PositiveInfinitySymbol

7 ToString
8

9 [C#] public string PositiveInfinitySymbol {get; set;}

10 [C++] public: __property String* get_PositiveInfinitySymbol();public: __property
11 void set_PositiveInfinitySymbol(String*);

12 [VB] Public Property PositiveInfinitySymbol As String

13 [JScript] public function get PositiveInfinitySymbol() : String;public function set
14 PositiveInfinitySymbol(String);
15

16 *Description*

17 Indicates the **System.String** that represents positive infinity.

18 PositiveSign

19 ToString
20

21 [C#] public string PositiveSign {get; set;}

22 [C++] public: __property String* get_PositiveSign();public: __property void
23 set_PositiveSign(String*);

24 [VB] Public Property PositiveSign As String

25 [JScript] public function get PositiveSign() : String;public function set

PositiveSign(String);

Description

Indicates the **System.String** that denotes that the associated number is positive.

This property is used only for parsing numeric strings, not for formatting.

Clone

[C#] public object Clone();

[C++] public: __sealed Object* Clone();

[VB] NotOverridable Public Function Clone() As Object

[JScript] public function Clone() : Object;

Description

Creates a shallow copy of the **System.Globalization.NumberFormatInfo** instance.

Return Value: A new **System.Globalization.NumberFormatInfo** instance copied from the original **System.Globalization.NumberFormatInfo** instance.

The clone is writable even if the original instance is read-only; therefore, the properties of the clone can be modified with user-defined patterns.

GetFormat

[C#] public object GetFormat(Type formatType);

[C++] public: __sealed Object* GetFormat(Type* formatType);

[VB] NotOverridable Public Function GetFormat(ByVal formatType As Type) As

Object

[JScript] public function GetFormat(formatType : Type) : Object;

Description

Gets an object of the specified type that provides a number formatting service.

Return Value: The current instance of the

System.Globalization.NumberFormatInfo class, if *formatType* is the same as the type of the current instance; otherwise, **null**.

The Format(String, IFormatProvider) methods supported by the base data types invoke this method when the current instance is passed as the **System.IFormatProvider** parameter. This method implements **System.IFormatProvider.GetFormat(System.Type)**. The **System.Type** of the required formatting service.

GetInstance

[C#] public static NumberFormatInfo GetInstance(IFormatProvider formatProvider);

[C++] public: static NumberFormatInfo* GetInstance(IFormatProvider* formatProvider);

[VB] Public Shared Function GetInstance(ByVal formatProvider As IFormatProvider) As NumberFormatInfo

[JScript] public static function GetInstance(formatProvider : IFormatProvider) : NumberFormatInfo;

1
2 *Description*

3 Gets the **System.Globalization.NumberFormatInfo** instance associated
4 with the specified **System.IFormatProvider** .

5 *Return Value:* The **System.Globalization.NumberFormatInfo** instance
6 associated with the specified **System.IFormatProvider** .

7 This method uses the
8 **System.IFormatProvider.GetFormat(System.Type)** method of *formatProvider*
9 using **System.Globalization.NumberFormatInfo** as the *Type* parameter. If
10 *formatProvider* is **null** or if **System.IFormatProvider.GetFormat(System.Type)**
11 returns **null** , this method returns
12 **System.Globalization.NumberFormatInfo.CurrentInfo** . The
13 **System.IFormatProvider** used to get the
14 **System.Globalization.NumberFormatInfo** instance.

15 **ReadOnly**

16
17 [C#] public static NumberFormatInfo ReadOnly(NumberFormatInfo nfi);
18 [C++] public: static NumberFormatInfo* ReadOnly(NumberFormatInfo* nfi);
19 [VB] Public Shared Function ReadOnly(ByVal nfi As NumberFormatInfo) As
20 NumberFormatInfo
21 [JScript] public static function ReadOnly(nfi : NumberFormatInfo) :
22 NumberFormatInfo;

23
24 *Description*
25

1 Returns a read-only **System.Globalization.NumberFormatInfo** wrapper.

2 *Return Value:* A read-only **System.Globalization.NumberFormatInfo** wrapper
3 around *nfi* .

4 This wrapper prevents any modifications to *nfi* . The
5 **System.Globalization.NumberFormatInfo** to wrap.

6 NumberStyles enumeration (System.Globalization)

7 ToString

8
9
10 *Description*

11 Determines the styles permitted in numerical string arguments that are
12 passed to the **Parse** methods of the numeric base type classes.

13 The symbols to use for currency symbol, thousands separator, decimal
14 point indicator, and leading sign are specified by

15 **System.Globalization.NumberFormatInfo** .

16 ToString

17
18 [C#] public const NumberStyles AllowCurrencySymbol;

19 [C++] public: const NumberStyles AllowCurrencySymbol;

20 [VB] Public Const AllowCurrencySymbol As NumberStyles

21 [JScript] public var AllowCurrencySymbol : NumberStyles;

22
23 *Description*

Indicates that a currency symbol is allowed. Valid currency symbols are determined by the **System.Globalization.NumberFormatInfo.CurrencySymbol** property of **System.Globalization.NumberFormatInfo** .

ToString

[C#] public const NumberStyles AllowDecimalPoint;

[C++] public: const NumberStyles AllowDecimalPoint;

[VB] Public Const AllowDecimalPoint As NumberStyles

[JScript] public var AllowDecimalPoint : NumberStyles;

Description

Indicates that a decimal point is allowed. Valid decimal point characters are determined by the **System.Globalization.NumberFormatInfo.NumberDecimalSeparator** and **System.Globalization.NumberFormatInfo.CurrencyDecimalSeparator** properties of **System.Globalization.NumberFormatInfo** .

ToString

[C#] public const NumberStyles AllowExponent;

[C++] public: const NumberStyles AllowExponent;

[VB] Public Const AllowExponent As NumberStyles

[JScript] public var AllowExponent : NumberStyles;

Description

Indicates that an exponent is allowed. The format of the number should be {e|E} [{+|-}] *n* , where *n* is a number.

ToString

[C#] public const NumberStyles AllowHexSpecifier;

[C++] public: const NumberStyles AllowHexSpecifier;

[VB] Public Const AllowHexSpecifier As NumberStyles

[JScript] public var AllowHexSpecifier : NumberStyles;

Description

Indicates that hexadecimal numbers are allowed.

ToString

[C#] public const NumberStyles AllowLeadingSign;

[C++] public: const NumberStyles AllowLeadingSign;

[VB] Public Const AllowLeadingSign As NumberStyles

[JScript] public var AllowLeadingSign : NumberStyles;

Description

Indicates that a leading sign is allowed. Valid leading sign characters are determined by the **System.Globalization.NumberFormatInfo.PositiveSign** and **System.Globalization.NumberFormatInfo.NegativeSign** properties of **System.Globalization.NumberFormatInfo** .

ToString

[C#] public const NumberStyles AllowLeadingWhite;
[C++] public: const NumberStyles AllowLeadingWhite;
[VB] Public Const AllowLeadingWhite As NumberStyles
[JScript] public var AllowLeadingWhite : NumberStyles;

Description

Indicates that a leading white-space character is allowed. Valid white-space characters have the Unicode values U+0009, U+000A, U+000B, U+000C, U+000D, and U+0020.

ToString

[C#] public const NumberStyles AllowParentheses;
[C++] public: const NumberStyles AllowParentheses;
[VB] Public Const AllowParentheses As NumberStyles
[JScript] public var AllowParentheses : NumberStyles;

Description

Indicates that parentheses are allowed.

ToString

[C#] public const NumberStyles AllowThousands;
[C++] public: const NumberStyles AllowThousands;
[VB] Public Const AllowThousands As NumberStyles
[JScript] public var AllowThousands : NumberStyles;

Description

Indicates that group separators are allowed; for instance, separating the hundreds from the thousands. Valid group separator characters are determined by the **System.Globalization.NumberFormatInfo.NumberGroupSeparator** and **System.Globalization.NumberFormatInfo.CurrencyGroupSeparator** properties of **System.Globalization.NumberFormatInfo** and the number of digits in each group is determined by the **System.Globalization.NumberFormatInfo.NumberGroupSizes** and **System.Globalization.NumberFormatInfo.CurrencyGroupSizes** properties of **System.Globalization.NumberFormatInfo**.

ToString

```
[C#] public const NumberStyles AllowTrailingSign;
[C++] public: const NumberStyles AllowTrailingSign;
[VB] Public Const AllowTrailingSign As NumberStyles
[JScript] public var AllowTrailingSign : NumberStyles;
```

Description

Indicates that a trailing sign is allowed. Valid trailing sign characters are determined by the **System.Globalization.NumberFormatInfo.PositiveSign** and **System.Globalization.NumberFormatInfo.NegativeSign** properties of **System.Globalization.NumberFormatInfo**.

ToString

```

1
2 [C#] public const NumberStyles AllowTrailingWhite;
3 [C++] public: const NumberStyles AllowTrailingWhite;
4 [VB] Public Const AllowTrailingWhite As NumberStyles
5 [JScript] public var AllowTrailingWhite : NumberStyles;
6

```

Description

Indicates that trailing white-space character is allowed. Valid white-space characters have the Unicode values U+0009, U+000A, U+000B, U+000C, U+000D, and U+0020.

ToString

```

12
13 [C#] public const NumberStyles Any;
14 [C++] public: const NumberStyles Any;
15 [VB] Public Const Any As NumberStyles
16 [JScript] public var Any : NumberStyles;
17

```

Description

Indicates that all the AllowXXX bit styles are used. This is a composite number style.

ToString

```

22
23 [C#] public const NumberStyles Currency;
24 [C++] public: const NumberStyles Currency;
25 [VB] Public Const Currency As NumberStyles

```

1 [JScript] public var Currency : NumberStyles;

3 *Description*

4 Indicates that all styles except AllowExponent are used. This is a composite
5 number style.

6 ToString

8 [C#] public const NumberStyles Float;

9 [C++] public: const NumberStyles Float;

10 [VB] Public Const Float As NumberStyles

11 [JScript] public var Float : NumberStyles;

13 *Description*

14 Indicates that the AllowLeadingWhite, AllowTrailingWhite,
15 AllowLeadingSign, AllowDecimalPoint, and AllowExponent styles are used. This
16 is a composite number style.

17 ToString

19 [C#] public const NumberStyles HexNumber;

20 [C++] public: const NumberStyles HexNumber;

21 [VB] Public Const HexNumber As NumberStyles

22 [JScript] public var HexNumber : NumberStyles;

24 *Description*

Indicates that the AllowLeadingWhite, AllowTrailingWhite, and AllowHexSpecifier styles are used. This is a composite number style.

ToString

[C#] public const NumberStyles Integer;

[C++] public: const NumberStyles Integer;

[VB] Public Const Integer As NumberStyles

[JScript] public var Integer : NumberStyles;

Description

Indicates that the AllowLeadingWhite, AllowTrailingWhite, and AllowLeadingSign styles are used. This is a composite number style.

ToString

[C#] public const NumberStyles None;

[C++] public: const NumberStyles None;

[VB] Public Const None As NumberStyles

[JScript] public var None : NumberStyles;

Description

Indicates that none of the bit styles are allowed.

ToString

[C#] public const NumberStyles Number;

[C++] public: const NumberStyles Number;

1 [VB] Public Const Number As NumberStyles

2 [JScript] public var Number : NumberStyles;

3
4 *Description*

5 Indicates that the AllowLeadingWhite, AllowTrailingWhite,
6 AllowLeadingSign, AllowTrailingSign, AllowDecimalPoint, and AllowThousands
7 styles are used. This is a composite number style.

8 RegionInfo class (System.Globalization)

9 ToString

10
11
12 *Description*

13 Contains information about the country/region.

14 In contrast to **System.Globalization.CultureInfo** ,
15 **System.Globalization.RegionInfo** does not represent preferences of the user and
16 does not depend on the user's language or culture.

17 RegionInfo

18 *Example Syntax:*

19 ToString

20
21 [C#] public RegionInfo(int culture);

22 [C++] public: RegionInfo(int culture);

23 [VB] Public Sub New(ByVal culture As Integer)

24 [JScript] public function RegionInfo(culture : int);

Description

Initializes a new instance of the **System.Globalization.RegionInfo** class based on the country/region associated with the specified culture identifier.

The culture identifier is mapped to the corresponding National Language Support (NLS) locale identifier. A list of culture identifiers is provided in the **System.Globalization.CultureInfo** class topic. A culture identifier.

RegionInfo

Example Syntax:

ToString

[C#] public RegionInfo(string name);

[C++] public: RegionInfo(String* name);

[VB] Public Sub New(ByVal name As String)

[JScript] public function RegionInfo(name : String); Initializes a new instance of the **System.Globalization.RegionInfo** class.

Description

Initializes a new instance of the **System.Globalization.RegionInfo** class based on the country/region specified by name.

The **System.Globalization.RegionInfo** name is one of the two-letter codes defined in ISO 3166 for country/region. A **System.String** containing one of the two-letter codes defined in ISO 3166 for country/region.

CurrencySymbol

ToString


```

1
2 [C#] public virtual string CurrencySymbol {get;}
3 [C++] public: __property virtual String* get_CurrencySymbol();
4 [VB] Overridable Public ReadOnly Property CurrencySymbol As String
5 [JScript] public function get CurrencySymbol() : String;
6

```

7 *Description*

8 Gets the currency symbol associated with the country/region.

9 For example, the currency symbol for the United States is "\$".

10 CurrentRegion

11 ToString

```

12
13 [C#] public static RegionInfo CurrentRegion {get;}
14 [C++] public: __property static RegionInfo* get_CurrentRegion();
15 [VB] Public Shared ReadOnly Property CurrentRegion As RegionInfo
16 [JScript] public static function get CurrentRegion() : RegionInfo;
17

```

18 *Description*

19 Gets the **System.Globalization.RegionInfo** instance that represents the
20 country/region used by the current thread.

21 The value of this property is based on the locale selected through the
22 Regional and Language Options (or Regional Options or Regional Settings) applet
23 in Windows Control Panel. However, that information can change during the life
24 of the **System.AppDomain** . The **System.Globalization.RegionInfo** class does
25 not detect changes in the system settings automatically.

1 DisplayName

2 ToString

3
4 [C#] public virtual string DisplayName {get;}

5 [C++] public: __property virtual String* get_DisplayName();

6 [VB] Overridable Public ReadOnly Property DisplayName As String

7 [JScript] public function get DisplayName() : String;

8
9 *Description*

10 Gets the full name of the country/region in the localized language of the
11 .NET Framework.

12 For example, if the .NET Framework English version is installed, the
13 United States is "United States". If the .NET Framework Spanish version is
14 installed, regardless of the language that the system is set to display, the
15 country/region name is displayed in Spanish; therefore, the United States is
16 "Estados Unidos".

17 EnglishName

18 ToString

19
20 [C#] public virtual string EnglishName {get;}

21 [C++] public: __property virtual String* get_EnglishName();

22 [VB] Overridable Public ReadOnly Property EnglishName As String

23 [JScript] public function get EnglishName() : String;

24
25 *Description*

Gets the full name of the country/region in English.

For example, the United States is "United States".

IsMetric

ToString

[C#] public virtual bool IsMetric {get;}

[C++] public: __property virtual bool get_IsMetric();

[VB] Overridable Public ReadOnly Property IsMetric As Boolean

[JScript] public function get IsMetric() : Boolean;

Description

Gets a value indicating whether the country/region uses the metric system for measurements.

ISOCurrencySymbol

ToString

[C#] public virtual string ISOCurrencySymbol {get;}

[C++] public: __property virtual String* get_ISOCurrencySymbol();

[VB] Overridable Public ReadOnly Property ISOCurrencySymbol As String

[JScript] public function get ISOCurrencySymbol() : String;

Description

Gets the three-character ISO 4217 currency symbol associated with the country/region.

1 A list of the three-character ISO 4217 currency symbols is provided in the
2 **System.Globalization.RegionInfo** class topic. For example, the ISO 4217
3 currency symbol for the United States dollar is "USD".

4 Name

5 ToString

6
7 [C#] public virtual string Name {get;}

8 [C++] public: __property virtual String* get_Name();

9 [VB] Overridable Public ReadOnly Property Name As String

10 [JScript] public function get Name() : String;

11
12 *Description*

13 Gets the two-letter code defined in ISO 3166 for the country/region.

14 The **System.Globalization.RegionInfo** name is one of the two-letter codes
15 defined in ISO 3166 for country/region. For example, the two-letter code for
16 United States is "US".

17 ThreeLetterISORegionName

18 ToString

19
20 [C#] public virtual string ThreeLetterISORegionName {get;}

21 [C++] public: __property virtual String* get_ThreeLetterISORegionName();

22 [VB] Overridable Public ReadOnly Property ThreeLetterISORegionName As
23 String

24 [JScript] public function get ThreeLetterISORegionName() : String;

1
2 *Description*

3 Gets the three-letter code defined in ISO 3166 for the country/region.

4 The **System.Globalization.RegionInfo.ThreeLetterISORegionName**
5 property contains one of the three-letter codes defined in ISO 3166 for
6 country/region. For example, the three-letter code for United States is "USA".

7 ThreeLetterWindowsRegionName

8 ToString

9
10 [C#] public virtual string ThreeLetterWindowsRegionName {get;}

11 [C++] public: __property virtual String* get_ThreeLetterWindowsRegionName();

12 [VB] Overridable Public ReadOnly Property ThreeLetterWindowsRegionName

13 As String

14 [JScript] public function get ThreeLetterWindowsRegionName() : String;

15
16 *Description*

17 Gets the Windows version of the three-letter code for the country/region of
18 this **System.Globalization.RegionInfo** .

19 For example, the three-letter code for United States is "USA".

20 TwoLetterISORegionName

21 ToString

22
23 [C#] public virtual string TwoLetterISORegionName {get;}

24 [C++] public: __property virtual String* get_TwoLetterISORegionName();

25 [VB] Overridable Public ReadOnly Property TwoLetterISORegionName As

String

[JScript] public function get TwoLetterISORegionName() : String;

Description

Gets the two-letter code defined in ISO 3166 for the country/region.

The **System.Globalization.RegionInfo** name is one of the two-letter codes defined in ISO 3166 for country/region. For example, the two-letter code for United States is "US".

Equals

[C#] public override bool Equals(object value);

[C++] public: bool Equals(Object* value);

[VB] Overrides Public Function Equals(ByVal value As Object) As Boolean

[JScript] public override function Equals(value : Object) : Boolean;

Description

Determines whether the specified **System.Object** is the same instance as the current **System.Globalization.RegionInfo** instance.

Return Value: **true** if the specified **System.Object** is the same instance as the current **System.Globalization.RegionInfo** instance; otherwise, **false** .

This method overrides **System.Object.Equals(System.Object)** . The **System.Object** to compare with the current **System.Globalization.RegionInfo** instance.

GetHashCode

1
2 [C#] public override int GetHashCode();

3 [C++] public: int GetHashCode();

4 [VB] Overrides Public Function GetHashCode() As Integer

5 [JScript] public override function GetHashCode() : int;

6
7 *Description*

8 Serves as a hash function for the current **System.Globalization.RegionInfo**
9 instance, suitable for use in hashing algorithms and data structures, such as a hash
10 table.

11 *Return Value:* A hash code for the current **System.Globalization.RegionInfo**
12 instance.

13 This method overrides **System.Object.GetHashCode** .

14 *ToString*

15
16 [C#] public override string ToString();

17 [C++] public: String* ToString();

18 [VB] Overrides Public Function ToString() As String

19 [JScript] public override function ToString() : String;

20
21 *Description*

22 Returns a **System.String** containing the name of the current
23 **System.Globalization.RegionInfo** instance, which is one of the three-letter
24 country/region codes defined in ISO 3166.

25 *Return Value:* A **System.String** containing the name of the current

System.Globalization.RegionInfo , which is one of the three-letter country/region codes defined in ISO 3166.

This method overrides **System.Object.ToString** .

SortKey class (System.Globalization)

ToString

Description

Maps strings to their sort keys.

Each character in a string is given several categories of sort weights, including script, alphabetic, case, and diacritic weights. A sort key serves as the repository of these weights for a particular string. For example, a sort key might contain a string of alphabetic weights, followed by a string of case weights, and so on.

KeyData

ToString

[C#] public virtual byte[] KeyData {get;}

[C++] public: __property virtual unsigned char get_KeyData();

[VB] Overridable Public ReadOnly Property KeyData As Byte ()

[JScript] public function get KeyData() : Byte[];

Description

Gets the byte array representing the current **System.Globalization.SortKey** instance.

OriginalString

ToString

[C#] public virtual string OriginalString {get;}

[C++] public: __property virtual String* get_OriginalString();

[VB] Overridable Public ReadOnly Property OriginalString As String

[JScript] public function get OriginalString() : String;

Description

Gets the original string used to create the current

System.Globalization.SortKey instance.

Compare

[C#] public static int Compare(SortKey sortkey1, SortKey sortkey2);

[C++] public: static int Compare(SortKey* sortkey1, SortKey* sortkey2);

[VB] Public Shared Function Compare(ByVal sortkey1 As SortKey, ByVal
sortkey2 As SortKey) As Integer

[JScript] public static function Compare(sortkey1 : SortKey, sortkey2 : SortKey) :
int;

Description

Compares two sort keys.

Return Value: Value Condition Zero The two sort keys are equal. The first sort
key to compare. The second sort key to compare.

Equals

1
2 [C#] public override bool Equals(object value);

3 [C++] public: bool Equals(Object* value);

4 [VB] Overrides Public Function Equals(ByVal value As Object) As Boolean

5 [JScript] public override function Equals(value : Object) : Boolean;

6
7 *Description*

8 Determines whether the specified **System.Object** is the same instance as
9 the current **System.Globalization.SortKey** .

10 *Return Value:* **true** if the specified **System.Object** is the same instance as the
11 current **System.Globalization.SortKey** ; otherwise, **false** .

12 This method overrides **System.Object.Equals(System.Object)** . The
13 **System.Object** to compare with the current **System.Globalization.SortKey**.

14 **GetHashCode**

15
16 [C#] public override int GetHashCode();

17 [C++] public: int GetHashCode();

18 [VB] Overrides Public Function GetHashCode() As Integer

19 [JScript] public override function GetHashCode() : int;

20
21 *Description*

22 Serves as a hash function for the current **System.Globalization.SortKey**
23 instance, suitable for use in hashing algorithms and data structures, such as a hash
24 table.

1 *Return Value:* A hash code for the current **System.Globalization.SortKey**
2 instance.

3 This method overrides **System.Object.GetHashCode** .

4 ToString

5
6 [C#] public override string ToString();

7 [C++] public: String* ToString();

8 [VB] Overrides Public Function ToString() As String

9 [JScript] public override function ToString() : String;

10
11 *Description*

12 Returns a **System.String** that represents the current
13 **System.Globalization.SortKey** instance.

14 *Return Value:* A **System.String** that represents the current
15 **System.Globalization.SortKey** instance.

16 This method overrides **System.Object.ToString** .

17 StringInfo class (System.Globalization)

18 ToString

19
20
21 *Description*

22 Provides functionality to split a string into text elements and to iterate
23 through those text elements.

24 The .NET Framework defines a text element as a unit of text that is
25 displayed as a single character; that is, a grapheme. A text element can be a base

1 character, a surrogate pair, or a combining character sequence. The Unicode
2 Standard defines a surrogate pair as a coded character representation for a single
3 abstract character that consists of a sequence of two code units, where the first unit
4 of the pair is a high-surrogate and the second is a low-surrogate. The Unicode
5 Standard defines a combining character sequence as a combination of a base
6 character and one or more combining characters. A surrogate pair can represent a
7 base character or a combining character. For more information on surrogate pairs
8 and combining character sequences, see The Unicode Standard at
9 <http://www.unicode.org>.

10 StringInfo

11 *Example Syntax:*

12 ToString

13
14 [C#] public StringInfo();

15 [C++] public: StringInfo();

16 [VB] Public Sub New()

17 [JScript] public function StringInfo();

18 GetNextTextElement

19
20 [C#] public static string GetNextTextElement(string str);

21 [C++] public: static String* GetNextTextElement(String* str);

22 [VB] Public Shared Function GetNextTextElement(ByVal str As String) As String

23 [JScript] public static function GetNextTextElement(str : String) : String; Gets the
24 first text element in a specified string.
25

Description

Gets the first text element in a specified string.

Return Value: A **System.String** containing the first text element in *str* .

The .NET Framework defines a text element as a unit of text that is displayed as a single character; that is, a grapheme. A text element can be a base character, a surrogate pair, or a combining character sequence. The Unicode Standard defines a surrogate pair as a coded character representation for a single abstract character that consists of a sequence of two code units, where the first unit of the pair is a high-surrogate and the second is a low-surrogate. The Unicode Standard defines a combining character sequence as a combination of a base character and one or more combining characters. A surrogate pair can represent a base character or a combining character. For more information on surrogate pairs and combining character sequences, see The Unicode Standard at <http://www.unicode.org>. The **System.String** to get the text element from.

GetNextTextElement

[C#] public static string GetNextTextElement(string str, int index);

[C++] public: static String* GetNextTextElement(String* str, int index);

[VB] Public Shared Function GetNextTextElement(ByVal str As String, ByVal index As Integer) As String

[JScript] public static function GetNextTextElement(str : String, index : int) : String;

Description

1 Gets the text element at the specified index of the specified string.

2 *Return Value:* A **System.String** containing the text element at *index* of *str* .

3 The .NET Framework defines a text element as a unit of text that is
4 displayed as a single character; that is, a grapheme. A text element can be a base
5 character, a surrogate pair, or a combining character sequence. The Unicode
6 Standard defines a surrogate pair as a coded character representation for a single
7 abstract character that consists of a sequence of two code units, where the first unit
8 of the pair is a high-surrogate and the second is a low-surrogate. The Unicode
9 Standard defines a combining character sequence as a combination of a base
10 character and one or more combining characters. A surrogate pair can represent a
11 base character or a combining character. For more information on surrogate pairs
12 and combining character sequences, see The Unicode Standard at
13 <http://www.unicode.org>. The **System.String** to get the text element from. The
14 index at which the text element starts.

15 GetTextElementEnumerator

16
17 [C#] public static TextElementEnumerator GetTextElementEnumerator(string str);

18 [C++] public: static TextElementEnumerator*

19 GetTextElementEnumerator(String* str);

20 [VB] Public Shared Function GetTextElementEnumerator(ByVal str As String) As

21 TextElementEnumerator

22 [JScript] public static function GetTextElementEnumerator(str : String) :

23 TextElementEnumerator; Returns an enumerator that can iterate through the text
24 elements of a **System.String** .
25

Description

Returns an enumerator that can iterate through the text elements of the entire **System.String**.

Return Value: A **System.Globalization.TextElementEnumerator** for the entire **System.String**.

The .NET Framework defines a text element as a unit of text that is displayed as a single character; that is, a grapheme. A text element can be a base character, a surrogate pair, or a combining character sequence. The Unicode Standard defines a surrogate pair as a coded character representation for a single abstract character that consists of a sequence of two code units, where the first unit of the pair is a high-surrogate and the second is a low-surrogate. The Unicode Standard defines a combining character sequence as a combination of a base character and one or more combining characters. A surrogate pair can represent a base character or a combining character. For more information on surrogate pairs and combining character sequences, see The Unicode Standard at <http://www.unicode.org>. The **System.String** to iterate through.

GetTextElementEnumerator

```
[C#] public static TextElementEnumerator GetTextElementEnumerator(string str,  
int index);
```

```
[C++] public: static TextElementEnumerator*  
GetTextElementEnumerator(String* str, int index);
```

```
[VB] Public Shared Function GetTextElementEnumerator(ByVal str As String,  
ByVal index As Integer) As TextElementEnumerator
```

1 [JScript] public static function GetTextElementEnumerator(str : String, index : int)
2 : TextElementEnumerator;
3

4 *Description*

5 Returns an enumerator that can iterate through the text elements of the
6 **System.String** starting at the specified index.

7 *Return Value:* A **System.Globalization.TextElementEnumerator** for the
8 **System.String** starting at the specified index.

9 The .NET Framework defines a text element as a unit of text that is
10 displayed as a single character; that is, a grapheme. A text element can be a base
11 character, a surrogate pair, or a combining character sequence. The Unicode
12 Standard defines a surrogate pair as a coded character representation for a single
13 abstract character that consists of a sequence of two code units, where the first unit
14 of the pair is a high-surrogate and the second is a low-surrogate. The Unicode
15 Standard defines a combining character sequence as a combination of a base
16 character and one or more combining characters. A surrogate pair can represent a
17 base character or a combining character. For more information on surrogate pairs
18 and combining character sequences, see The Unicode Standard at
19 <http://www.unicode.org>. The **System.String** to iterate through. The index at which
20 to start iterating.

21 **ParseCombiningCharacters**

22
23 [C#] public static int[] ParseCombiningCharacters(string str);

24 [C++] public: static int ParseCombiningCharacters(String* str) __gc[];

25 [VB] Public Shared Function ParseCombiningCharacters(ByVal str As String) As

Integer()

[JScript] public static function ParseCombiningCharacters(str : String) : int[];

Description

Returns the indexes of each base character, high-surrogate, or control character within the specified string.

Return Value: An array of integers that contains the indexes of each base character, high-surrogate, or control character within the specified string.

The Unicode Standard defines a surrogate pair as a coded character representation for a single abstract character that consists of a sequence of two code units, where the first unit of the pair is a high-surrogate and the second is a low-surrogate. A high-surrogate is a Unicode code point in the range U+D800 through U+DBFF and a low-surrogate is a Unicode code point in the range U+DC00 through U+DFFF. The **System.String** to search.

TaiwanCalendar class (System.Globalization)

ToString

Description

Represents the Taiwanese calendar.

The Taiwanese calendar works exactly like the Gregorian calendar, except that the year and era are different.

TaiwanCalendar

Example Syntax:

ToString

```

1
2 [C#] public TaiwanCalendar();
3 [C++] public: TaiwanCalendar();
4 [VB] Public Sub New()
5 [JScript] public function TaiwanCalendar();
6

```

Description

Initializes a new instance of the **System.Globalization.TaiwanCalendar** class.

Eras

ToString

```

13 [C#] public override int[] Eras {get;}
14 [C++] public: __property virtual int get_Eras();
15 [VB] Overrides Public ReadOnly Property Eras As Integer ()
16 [JScript] public function get Eras() : int[];
17

```

Description

Gets the list of eras in the **System.Globalization.TaiwanCalendar** .

The **System.Globalization.TaiwanCalendar** class recognizes only the current era. This property always returns an array with only one element.

TwoDigitYearMax

ToString

```

25 [C#] public override int TwoDigitYearMax {get; set;}

```

```

1 [C++] public: __property virtual int get_TwoDigitYearMax();public: __property
2 virtual void set_TwoDigitYearMax(int);
3 [VB] Overrides Public Property TwoDigitYearMax As Integer
4 [JScript] public function get TwoDigitYearMax() : int;public function set
5 TwoDigitYearMax(int);
6

```

Description

Gets or sets the last year of a 100-year range that can be represented by a 2-digit year.

This property allows a 2-digit year to be properly translated to a 4-digit year. For example, in the Gregorian calendar, if this property is set to 2029, the 100-year range is from 1930 to 2029; therefore, a 2-digit value of 30 is interpreted as 1930, while a 2-digit value of 29 is interpreted as 2029.

AddMonths

```

16 [C#] public override DateTime AddMonths(DateTime time, int months);
17 [C++] public: DateTime AddMonths(DateTime time, int months);
18 [VB] Overrides Public Function AddMonths(ByVal time As DateTime, ByVal
19 months As Integer) As DateTime
20 [JScript] public override function AddMonths(time : DateTime, months : int) :
21 DateTime;
22

```

Description

Returns a **System.DateTime** that is the specified number of months away from the specified **System.DateTime**.

Return Value: The **System.DateTime** that results from adding the specified number of months to the specified **System.DateTime** .

The year part of the resulting **System.DateTime** is affected if the resulting month is beyond the last month of the current year. The day part of the resulting **System.DateTime** is also affected if the resulting day is not a valid day in the resulting month of the resulting year; it is changed to the last valid day in the resulting month of the resulting year. The time-of-day part of the resulting **System.DateTime** remains the same as the specified **System.DateTime** . The **System.DateTime** instance to add. The number of months to add.

AddYears

[C#] public override DateTime AddYears(DateTime time, int years);

[C++] public: DateTime AddYears(DateTime time, int years);

[VB] Overrides Public Function AddYears(ByVal time As DateTime, ByVal years As Integer) As DateTime

[JScript] public override function AddYears(time : DateTime, years : int) : DateTime;

Description

Returns a **System.DateTime** that is the specified number of years away from the specified **System.DateTime** .

Return Value: The **System.DateTime** that results from adding the specified number of years to the specified **System.DateTime** .

The day part of the resulting **System.DateTime** is affected if the resulting day is not a valid day in the resulting month of the resulting year; it is changed to

the last valid day in the resulting month of the resulting year. The time-of-day part of the resulting **System.DateTime** remains the same as the specified **System.DateTime**. The **System.DateTime** instance to add. The number of years to add.

GetDayOfMonth

[C#] public override int GetDayOfMonth(DateTime time);

[C++] public: int GetDayOfMonth(DateTime time);

[VB] Overrides Public Function GetDayOfMonth(ByVal time As DateTime) As

Integer

[JScript] public override function GetDayOfMonth(time : DateTime) : int;

Description

Gets the day of the month in the specified **System.DateTime**.

Return Value: An integer from 1 to 31 that represents the day of the month in *time*

. The **System.DateTime** instance to read.

GetDayOfWeek

[C#] public override DayOfWeek GetDayOfWeek(DateTime time);

[C++] public: DayOfWeek GetDayOfWeek(DateTime time);

[VB] Overrides Public Function GetDayOfWeek(ByVal time As DateTime) As

DayOfWeek

[JScript] public override function GetDayOfWeek(time : DateTime) :

DayOfWeek;

1
2 *Description*

3 Gets the day of the week in the specified **System.DateTime** .

4 *Return Value:* A **System.DayOfWeek** value that represents the day of the week in
5 *time* .

6 The **System.DayOfWeek** values are Sunday, Monday, Tuesday,
7 Wednesday, Thursday, Friday, and Saturday. The **System.DateTime** instance to
8 read.

9 **GetDayOfYear**

10
11 [C#] public override int GetDayOfYear(DateTime time);

12 [C++] public: int GetDayOfYear(DateTime time);

13 [VB] Overrides Public Function GetDayOfYear(ByVal time As DateTime) As
14 Integer

15 [JScript] public override function GetDayOfYear(time : DateTime) : int;

16
17 *Description*

18 Gets the day of the year in the specified **System.DateTime** .

19 *Return Value:* An integer from 1 to 366 that represents the day of the year in *time* .

20 The **System.DateTime** instance to read.

21 **GetDaysInMonth**

22
23 [C#] public override int GetDaysInMonth(int year, int month, int era);

24 [C++] public: int GetDaysInMonth(int year, int month, int era);

25 [VB] Overrides Public Function GetDaysInMonth(ByVal year As Integer, ByVal

month As Integer, ByVal era As Integer) As Integer

[JScript] public override function GetDaysInMonth(year : int, month : int, era : int) : int; Gets the number of days in the specified month.

Description

Gets the number of days in the month specified by the *year*, *month*, and *era* parameters.

Return Value: The number of days in the specified month in the specified year in the specified era.

For example, this method might return 28 or 29 for February (*month* = 2), depending on whether *year* is a leap year. An integer that represents the year. An integer that represents the month. An integer that represents the era.

GetDaysInYear

[C#] public override int GetDaysInYear(int year, int era);

[C++] public: int GetDaysInYear(int year, int era);

[VB] Overrides Public Function GetDaysInYear(ByVal year As Integer, ByVal era As Integer) As Integer

[JScript] public override function GetDaysInYear(year : int, era : int) : int; Gets the number of days in the specified year.

Description

Gets the number of days in the year specified by the *year* and *era* parameters.

Return Value: The number of days in the specified year in the specified era.

For example, this method might return 365 or 366, depending on whether *year* is a leap year. An integer that represents the year. An integer that represents the era.

GetEra

[C#] public override int GetEra(DateTime time);

[C++] public: int GetEra(DateTime time);

[VB] Overrides Public Function GetEra(ByVal time As DateTime) As Integer

[JScript] public override function GetEra(time : DateTime) : int;

Description

Gets the era in the specified **System.DateTime**.

Return Value: An integer that represents the era in *time*.

The **System.Globalization.TaiwanCalendar** class recognizes only the current era. The **System.DateTime** instance to read.

GetMonth

[C#] public override int GetMonth(DateTime time);

[C++] public: int GetMonth(DateTime time);

[VB] Overrides Public Function GetMonth(ByVal time As DateTime) As Integer

[JScript] public override function GetMonth(time : DateTime) : int;

Description

Gets the month in the specified **System.DateTime** .

Return Value: An integer between 1 and 12 that represents the month in *time* . The **System.DateTime** instance to read.

GetMonthsInYear

[C#] public override int GetMonthsInYear(int year, int era);

[C++] public: int GetMonthsInYear(int year, int era);

[VB] Overrides Public Function GetMonthsInYear(ByVal year As Integer, ByVal era As Integer) As Integer

[JScript] public override function GetMonthsInYear(year : int, era : int) : int; Gets the number of months in the specified year.

Description

Gets the number of months in the year specified by the *year* and *era* parameters.

Return Value: The number of months in the specified year in the specified era. An integer that represents the year. An integer that represents the era.

GetYear

[C#] public override int GetYear(DateTime time);

[C++] public: int GetYear(DateTime time);

[VB] Overrides Public Function GetYear(ByVal time As DateTime) As Integer

[JScript] public override function GetYear(time : DateTime) : int;

Description

Gets the year in the specified **System.DateTime** .

Return Value: An integer between 1 and 9999 that represents the year in *time* . The **System.DateTime** instance to read.

IsLeapDay

[C#] public override bool IsLeapDay(int year, int month, int day, int era);

[C++] public: bool IsLeapDay(int year, int month, int day, int era);

[VB] Overrides Public Function IsLeapDay(ByVal year As Integer, ByVal month As Integer, ByVal day As Integer, ByVal era As Integer) As Boolean

[JScript] public override function IsLeapDay(year : int, month : int, day : int, era : int) : Boolean; Determines whether the specified day is a leap day.

Description

Determines whether the date specified by the *year* , *month* , *day* , and *era* parameters is a leap day.

Return Value: **true** if the specified day is a leap day; otherwise, **false** .

Leap years in the Taiwanese calendar correspond to the same leap years in the Gregorian calendar. A common year has 365 days and a leap year has 366 days. An integer that represents the year. An integer that represents the month. An integer that represents the day. An integer that represents the era.

IsLeapMonth

[C#] public override bool IsLeapMonth(int year, int month, int era);

[C++] public: bool IsLeapMonth(int year, int month, int era);

[VB] Overrides Public Function IsLeapMonth(ByVal year As Integer, ByVal

month As Integer, ByVal era As Integer) As Boolean

[JScript] public override function IsLeapMonth(year : int, month : int, era : int) :

Boolean; Determines whether the specified month is a leap month.

Description

Determines whether the month specified by the *year* , *month* , and *era* parameters is a leap month.

Return Value: This method always returns **false** , unless overridden by a derived class.

Leap years in the Taiwanese calendar correspond to the same leap years in the Gregorian calendar. A common year has 365 days and a leap year has 366 days. An integer that represents the year. An integer that represents the month. An integer that represents the era.

IsLeapYear

[C#] public override bool IsLeapYear(int year, int era);

[C++] public: bool IsLeapYear(int year, int era);

[VB] Overrides Public Function IsLeapYear(ByVal year As Integer, ByVal era As Integer) As Boolean

[JScript] public override function IsLeapYear(year : int, era : int) : Boolean;

Determines whether the specified year is a leap year.

Description

Determines whether the year specified by the *year* and *era* parameters is a leap year.

Return Value: **true** if the specified year is a leap year; otherwise, **false** .

Leap years in the Taiwanese calendar correspond to the same leap years in the Gregorian calendar. A common year has 365 days and a leap year has 366 days. An integer that represents the year. An integer that represents the era.

ToDateTime

[C#] public override DateTime ToDateTime(int year, int month, int day, int hour, int minute, int second, int millisecond, int era);

[C++] public: DateTime ToDateTime(int year, int month, int day, int hour, int minute, int second, int millisecond, int era);

[VB] Overrides Public Function ToDateTime(ByVal year As Integer, ByVal month As Integer, ByVal day As Integer, ByVal hour As Integer, ByVal minute As Integer, ByVal second As Integer, ByVal millisecond As Integer, ByVal era As Integer) As DateTime

[JScript] public override function ToDateTime(year : int, month : int, day : int, hour : int, minute : int, second : int, millisecond : int, era : int) : DateTime; Returns a **System.DateTime** that is set to the specified date.

Description

Returns a **System.DateTime** that is set to the specified date and time in the specified era.

Return Value: The **System.DateTime** instance set to the specified date and time in the current era. An integer that represents the year. An integer that represents the

month. An integer that represents the day. An integer that represents the hour. An integer that represents the minute. An integer that represents the second. An integer that represents the millisecond. An integer that represents the era.

ToFourDigitYear

[C#] public override int ToFourDigitYear(int year);

[C++] public: int ToFourDigitYear(int year);

[VB] Overrides Public Function ToFourDigitYear(ByVal year As Integer) As Integer

[JScript] public override function ToFourDigitYear(year : int) : int;

Description

Converts the specified two-digit year to a four-digit year by using the **System.Globalization.TaiwanCalendar.TwoDigitYearMax** property to determine the appropriate century.

Return Value: An integer that contains the four-digit representation of *year* .

This method implements

System.Globalization.Calendar.ToFourDigitYear(System.Int32) . A two-digit integer that represents the year to convert.

TextElementEnumerator class (System.Globalization)

ToString

Description

Enumerates the text elements of a **System.String** .

The .NET Framework defines a text element as a unit of text that is displayed as a single character; that is, a grapheme. A text element can be a base character, a surrogate pair, or a combining character sequence. The Unicode Standard defines a surrogate pair as a coded character representation for a single abstract character that consists of a sequence of two code units, where the first unit of the pair is a high-surrogate and the second is a low-surrogate. The Unicode Standard defines a combining character sequence as a combination of a base character and one or more combining characters. A surrogate pair can represent a base character or a combining character. For more information on surrogate pairs and combining character sequences, see The Unicode Standard at <http://www.unicode.org>.

Current

ToString

```
[C#] public object Current {get;}
[C++] public: __property Object* get_Current();
[VB] Public ReadOnly Property Current As Object
[JScript] public function get Current() : Object;
```

Description

Gets the current text element in the **System.String**.

The .NET Framework defines a text element as a unit of text that is displayed as a single character; that is, a grapheme. A text element can be a base character, a surrogate pair, or a combining character sequence. The Unicode Standard defines a surrogate pair as a coded character representation for a single

abstract character that consists of a sequence of two code units, where the first unit of the pair is a high-surrogate and the second is a low-surrogate. The Unicode Standard defines a combining character sequence as a combination of a base character and one or more combining characters. A surrogate pair can represent a base character or a combining character. For more information on surrogate pairs and combining character sequences, see The Unicode Standard at <http://www.unicode.org>.

ElementIndex

ToString

[C#] public int ElementIndex {get;}

[C++] public: __property int get_ElementIndex();

[VB] Public ReadOnly Property ElementIndex As Integer

[JScript] public function get ElementIndex() : int;

Description

Gets the index of the text element that the enumerator is currently positioned over.

The .NET Framework defines a text element as a unit of text that is displayed as a single character; that is, a grapheme. A text element can be a base character, a surrogate pair, or a combining character sequence. The Unicode Standard defines a surrogate pair as a coded character representation for a single abstract character that consists of a sequence of two code units, where the first unit of the pair is a high-surrogate and the second is a low-surrogate. The Unicode Standard defines a combining character sequence as a combination of a base

character and one or more combining characters. A surrogate pair can represent a base character or a combining character. For more information on surrogate pairs and combining character sequences, see The Unicode Standard at <http://www.unicode.org>.

GetTextElement

```
[C#] public string GetTextElement();  
[C++] public: String* GetTextElement();  
[VB] Public Function GetTextElement() As String  
[JScript] public function GetTextElement() : String;
```

Description

Gets the current text element in the **System.String**.

Return Value: A **System.String** instance containing the current text element in the **System.String**.

The .NET Framework defines a text element as a unit of text that is displayed as a single character; that is, a grapheme. A text element can be a base character, a surrogate pair, or a combining character sequence. The Unicode Standard defines a surrogate pair as a coded character representation for a single abstract character that consists of a sequence of two code units, where the first unit of the pair is a high-surrogate and the second is a low-surrogate. The Unicode Standard defines a combining character sequence as a combination of a base character and one or more combining characters. A surrogate pair can represent a base character or a combining character. For more information on surrogate pairs

and combining character sequences, see The Unicode Standard at
<http://www.unicode.org>.

MoveNext

[C#] public bool MoveNext();

[C++] public: __sealed bool MoveNext();

[VB] NotOverridable Public Function MoveNext() As Boolean

[JScript] public function MoveNext() : Boolean;

Description

Advances the enumerator to the next text element of the **System.String** .
Return Value: **true** if the enumerator was successfully advanced to the next text element; **false** if the enumerator has passed the end of the **System.String** .

The .NET Framework defines a text element as a unit of text that is displayed as a single character; that is, a grapheme. A text element can be a base character, a surrogate pair, or a combining character sequence. The Unicode Standard defines a surrogate pair as a coded character representation for a single abstract character that consists of a sequence of two code units, where the first unit of the pair is a high-surrogate and the second is a low-surrogate. The Unicode Standard defines a combining character sequence as a combination of a base character and one or more combining characters. A surrogate pair can represent a base character or a combining character. For more information on surrogate pairs and combining character sequences, see The Unicode Standard at
<http://www.unicode.org>.

Reset

[C#] public void Reset();
[C++] public: __sealed void Reset();
[VB] NotOverridable Public Sub Reset()
[JScript] public function Reset();

Description

Sets the enumerator to its initial position, which is before the first text element in the **System.String**.

The .NET Framework defines a text element as a unit of text that is displayed as a single character; that is, a grapheme. A text element can be a base character, a surrogate pair, or a combining character sequence. The Unicode Standard defines a surrogate pair as a coded character representation for a single abstract character that consists of a sequence of two code units, where the first unit of the pair is a high-surrogate and the second is a low-surrogate. The Unicode Standard defines a combining character sequence as a combination of a base character and one or more combining characters. A surrogate pair can represent a base character or a combining character. For more information on surrogate pairs and combining character sequences, see The Unicode Standard at <http://www.unicode.org>.

TextInfo class (System.Globalization)

ToString

Description

1 Defines properties and behaviors, such as casing, that are specific to a
2 writing system.

3 A writing system is the collection of scripts and orthographic rules required
4 to represent a language as text.

5 `ANSICodePage`

6 `ToString`

7
8 [C#] `public virtual int ANSICodePage {get;}`

9 [C++] `public: __property virtual int get_ANSICodePage();`

10 [VB] `Overridable Public ReadOnly Property ANSICodePage As Integer`

11 [JScript] `public function get ANSICodePage() : int;`

12
13 *Description*

14 Gets the American National Standards Institute (ANSI) code page used by
15 the writing system represented by the **System.Globalization.TextInfo** instance.

16 `EBCDICCodePage`

17 `ToString`

18
19 [C#] `public virtual int EBCDICCodePage {get;}`

20 [C++] `public: __property virtual int get_EBCDICCodePage();`

21 [VB] `Overridable Public ReadOnly Property EBCDICCodePage As Integer`

22 [JScript] `public function get EBCDICCodePage() : int;`

23
24 *Description*

Gets the Extended Binary Coded Decimal Interchange Code (EBCDIC)
code page used by the writing system represented by the
System.Globalization.TextInfo instance.

ListSeparator

ToString

[C#] public virtual string ListSeparator {get;}

[C++] public: __property virtual String* get_ListSeparator();

[VB] Overridable Public ReadOnly Property ListSeparator As String

[JScript] public function get ListSeparator() : String;

Description

Gets the **System.String** that separates items in a list.

The default for the invariant culture is ",".

MacCodePage

ToString

[C#] public virtual int MacCodePage {get;}

[C++] public: __property virtual int get_MacCodePage();

[VB] Overridable Public ReadOnly Property MacCodePage As Integer

[JScript] public function get MacCodePage() : int;

Description

Gets the Macintosh code page used by the writing system represented by
the **System.Globalization.TextInfo** instance.

OEMCodePage

ToString

[C#] public virtual int OEMCodePage {get;}

[C++] public: __property virtual int get _OEMCodePage();

[VB] Overridable Public ReadOnly Property OEMCodePage As Integer

[JScript] public function get OEMCodePage() : int;

Description

Gets the original equipment manufacturer (OEM) code page used by the writing system represented by the **System.Globalization.TextInfo** instance.

Equals

[C#] public override bool Equals(object obj);

[C++] public: bool Equals(Object* obj);

[VB] Overrides Public Function Equals(ByVal obj As Object) As Boolean

[JScript] public override function Equals(obj : Object) : Boolean;

Description

Determines whether the specified **System.Object** represents the same writing system as the current **System.Globalization.TextInfo** .

Return Value: **true** if the specified **System.Object** represents the same writing system as the current **System.Globalization.TextInfo** ; otherwise, **false** .

This method overrides **System.Object.Equals(System.Object)** . The **System.Object** to compare with the current **System.Globalization.TextInfo**.

GetHashCode

[C#] public override int GetHashCode();

[C++] public: int GetHashCode();

[VB] Overrides Public Function GetHashCode() As Integer

[JScript] public override function GetHashCode() : int;

Description

Serves as a hash function for the current **System.Globalization.TextInfo** instance, suitable for use in hashing algorithms and data structures, such as a hash table.

Return Value: A hash code for the current **System.Globalization.TextInfo** instance.

This method overrides **System.Object.GetHashCode** .

IDeserializationCallback.OnDeserialization

[C#] void IDeserializationCallback.OnDeserialization(object sender);

[C++] void IDeserializationCallback::OnDeserialization(Object* sender);

[VB] Sub OnDeserialization(ByVal sender As Object) Implements

IDeserializationCallback.OnDeserialization

[JScript] function IDeserializationCallback.OnDeserialization(sender : Object);

ToLower

[C#] public virtual char ToLower(char c);

[C++] public: virtual __wchar_t ToLower(__wchar_t c);

[VB] Overridable Public Function ToLower(ByVal c As Char) As Char
[JScript] public function ToLower(c : Char) : Char; Converts the specified character or string to lowercase.

Description

Converts the specified character to lowercase.

Return Value: The specified character converted to lowercase.

Casing semantics depend on the culture in use. If using the invariant culture, the casing semantics are not culture-sensitive. If using a specific culture, the casing semantics are sensitive to that culture. The character to convert to lowercase.

ToLower

[C#] public virtual string ToLower(string str);
[C++] public: virtual String* ToLower(String* str);
[VB] Overridable Public Function ToLower(ByVal str As String) As String
[JScript] public function ToLower(str : String) : String;

Description

Converts the specified string to lowercase.

Return Value: The specified string converted to lowercase.

The returned string might differ in length from the input string. For more information on casing, refer to the Unicode Technical Report #21 "Case Mappings," published by the Unicode Consortium (<http://www.unicode.org>). The current implementation preserves the length of the string; however, this behavior

is not guaranteed and could change in future implementations. The string to convert to lowercase.

ToString

[C#] public override string ToString();

[C++] public: String* ToString();

[VB] Overrides Public Function ToString() As String

[JScript] public override function ToString() : String;

Description

Returns a **System.String** that represents the current **System.Globalization.TextInfo** instance.

Return Value: A **System.String** that represents the current **System.Globalization.TextInfo** instance.

This method overrides **System.Object.ToString**.

ToTitleCase

[C#] public string ToTitleCase(string str);

[C++] public: String* ToTitleCase(String* str);

[VB] Public Function ToTitleCase(ByVal str As String) As String

[JScript] public function ToTitleCase(str : String) : String;

Description

Converts the specified string to titlecase.

Return Value: The specified string converted to titlecase.

Generally, title casing converts the first character of a word to uppercase and converts the rest of the letters to lowercase. Words that are selected for title casing is dependent on the language. The string to convert to titlecase.

ToUpper

[C#] public virtual char ToUpper(char c);

[C++] public: virtual __wchar_t ToUpper(__wchar_t c);

[VB] Overridable Public Function ToUpper(ByVal c As Char) As Char

[JScript] public function ToUpper(c : Char) : Char; Converts the specified character or string to uppercase.

Description

Converts the specified character to uppercase.

Return Value: The specified character converted to uppercase.

Casing semantics depend on the culture in use. If using the invariant culture, the casing semantics are not culture-sensitive. If using a specific culture, the casing semantics are sensitive to that culture. The character to convert to uppercase.

ToUpper

[C#] public virtual string ToUpper(string str);

[C++] public: virtual String* ToUpper(String* str);

[VB] Overridable Public Function ToUpper(ByVal str As String) As String

[JScript] public function ToUpper(str : String) : String;

1
2 *Description*

3 Converts the specified string to uppercase.

4 *Return Value:* The specified string converted to uppercase.

5 The returned string might differ in length from the input string. For more
6 information on casing, refer to the Unicode Technical Report #21 "Case
7 Mappings," published by the Unicode Consortium (<http://www.unicode.org>). The
8 current implementation preserves the length of the string; however this behavior
9 might change in future versions of the .NET Framework. The string to convert to
10 uppercase.

11 ThaiBuddhistCalendar class (System.Globalization)

12 ToUpper

13
14
15 *Description*

16 Represents the Thai Buddhist calendar.

17 The Thai Buddhist calendar works exactly like the Gregorian calendar,
18 except that the year and era are different.

19 ToUpper

20
21 [C#] public const int ThaiBuddhistEra;

22 [C++] public: const int ThaiBuddhistEra;

23 [VB] Public Const ThaiBuddhistEra As Integer

24 [JScript] public var ThaiBuddhistEra : int;

1
2 *Description*

3 Represents the current era.

4 The **System.Globalization.ThaiBuddhistCalendar** class recognizes only
5 the current era. This field always returns 1.

6 ThaiBuddhistCalendar

7 *Example Syntax:*

8 ToUpper

9
10 [C#] public ThaiBuddhistCalendar();

11 [C++] public: ThaiBuddhistCalendar();

12 [VB] Public Sub New()

13 [JScript] public function ThaiBuddhistCalendar();

14
15 *Description*

16 Initializes a new instance of the
17 **System.Globalization.ThaiBuddhistCalendar** class.

18 Eras

19 ToUpper

20
21 [C#] public override int[] Eras {get;}

22 [C++] public: __property virtual int get_Eras();

23 [VB] Overrides Public ReadOnly Property Eras As Integer ()

24 [JScript] public function get Eras() : int[];

Description

Gets the list of eras in the **System.Globalization.ThaiBuddhistCalendar**.

The **System.Globalization.ThaiBuddhistCalendar** class recognizes only the current era. This property always returns an array with only one element.

TwoDigitYearMax

ToUpper

```
[C#] public override int TwoDigitYearMax {get; set;}
```

```
[C++] public: __property virtual int get_TwoDigitYearMax();public: __property  
virtual void set_TwoDigitYearMax(int);
```

```
[VB] Overrides Public Property TwoDigitYearMax As Integer
```

```
[JScript] public function get TwoDigitYearMax() : int;public function set  
TwoDigitYearMax(int);
```

Description

Gets or sets the last year of a 100-year range that can be represented by a 2-digit year.

This property allows a 2-digit year to be properly translated to a 4-digit year. For example, in the Gregorian calendar, if this property is set to 2029, the 100-year range is from 1930 to 2029; therefore, a 2-digit value of 30 is interpreted as 1930, while a 2-digit value of 29 is interpreted as 2029.

AddMonths

```
[C#] public override DateTime AddMonths(DateTime time, int months);
```

1 [C++] public: DateTime AddMonths(DateTime time, int months);

2 [VB] Overrides Public Function AddMonths(ByVal time As DateTime, ByVal
3 months As Integer) As DateTime

4 [JScript] public override function AddMonths(time : DateTime, months : int) :
5 DateTime;

6 7 *Description*

8 Returns a **System.DateTime** that is the specified number of months away
9 from the specified **System.DateTime** .

10 *Return Value:* The **System.DateTime** that results from adding the specified
11 number of months to the specified **System.DateTime** .

12 The year part of the resulting **System.DateTime** is affected if the resulting
13 month is beyond the last month of the current year. The day part of the resulting
14 **System.DateTime** is also affected if the resulting day is not a valid day in the
15 resulting month of the resulting year; it is changed to the last valid day in the
16 resulting month of the resulting year. The time-of-day part of the resulting
17 **System.DateTime** remains the same as the specified **System.DateTime** . The
18 **System.DateTime** instance to add. The number of months to add.

19 *AddYears*

20
21 [C#] public override DateTime AddYears(DateTime time, int years);

22 [C++] public: DateTime AddYears(DateTime time, int years);

23 [VB] Overrides Public Function AddYears(ByVal time As DateTime, ByVal years
24 As Integer) As DateTime

25 [JScript] public override function AddYears(time : DateTime, years : int) :

DateTime;

Description

Returns a **System.DateTime** that is the specified number of years away from the specified **System.DateTime**.

Return Value: The **System.DateTime** that results from adding the specified number of years to the specified **System.DateTime**.

The day part of the resulting **System.DateTime** is affected if the resulting day is not a valid day in the resulting month of the resulting year; it is changed to the last valid day in the resulting month of the resulting year. The time-of-day part of the resulting **System.DateTime** remains the same as the specified **System.DateTime**. The **System.DateTime** instance to add. The number of years to add.

GetDayOfMonth

[C#] public override int GetDayOfMonth(DateTime time);

[C++] public: int GetDayOfMonth(DateTime time);

[VB] Overrides Public Function GetDayOfMonth(ByVal time As DateTime) As Integer

[JScript] public override function GetDayOfMonth(time : DateTime) : int;

Description

Gets the day of the month in the specified **System.DateTime**.

Return Value: An integer from 1 to 31 that represents the day of the month in *time*. The **System.DateTime** instance to read.

GetDayOfWeek

[C#] public override DayOfWeek GetDayOfWeek(DateTime time);
[C++] public: DayOfWeek GetDayOfWeek(DateTime time);
[VB] Overrides Public Function GetDayOfWeek(ByVal time As DateTime) As
DayOfWeek
[JScript] public override function GetDayOfWeek(time : DateTime) :
DayOfWeek;

Description

Gets the day of the week in the specified **System.DateTime** .

Return Value: A **System.DayOfWeek** value that represents the day of the week in
time .

The **System.DayOfWeek** values are Sunday, Monday, Tuesday,
Wednesday, Thursday, Friday, and Saturday. The **System.DateTime** instance to
read.

GetDayOfYear

[C#] public override int GetDayOfYear(DateTime time);
[C++] public: int GetDayOfYear(DateTime time);
[VB] Overrides Public Function GetDayOfYear(ByVal time As DateTime) As
Integer
[JScript] public override function GetDayOfYear(time : DateTime) : int;

Description

Gets the day of the year in the specified **System.DateTime** .

Return Value: An integer from 1 to 366 that represents the day of the year in *time* .

The **System.DateTime** instance to read.

GetDaysInMonth

[C#] public override int GetDaysInMonth(int year, int month, int era);

[C++] public: int GetDaysInMonth(int year, int month, int era);

[VB] Overrides Public Function GetDaysInMonth(ByVal year As Integer, ByVal month As Integer, ByVal era As Integer) As Integer

[JScript] public override function GetDaysInMonth(year : int, month : int, era : int) : int; Gets the number of days in the specified month.

Description

Gets the number of days in the month specified by the *year* , *month* , and *era* parameters.

Return Value: The number of days in the specified month in the specified year in the specified era.

For example, this method might return 28 or 29 for February (*month* = 2), depending on whether *year* is a leap year. An integer that represents the year. An integer that represents the month. An integer that represents the era.

GetDaysInYear

[C#] public override int GetDaysInYear(int year, int era);

[C++] public: int GetDaysInYear(int year, int era);

[VB] Overrides Public Function GetDaysInYear(ByVal year As Integer, ByVal

era As Integer) As Integer

[JScript] public override function GetDaysInYear(year : int, era : int) : int; Gets the number of days in the specified year.

Description

Gets the number of days in the year specified by the *year* and *era* parameters.

Return Value: The number of days in the specified year in the specified era.

For example, this method might return 365 or 366, depending on whether *year* is a leap year. An integer that represents the year. An integer that represents the era.

GetEra

[C#] public override int GetEra(DateTime time);

[C++] public: int GetEra(DateTime time);

[VB] Overrides Public Function GetEra(ByVal time As DateTime) As Integer

[JScript] public override function GetEra(time : DateTime) : int;

Description

Gets the era in the specified **System.DateTime** .

Return Value: An integer that represents the era in *time* .

The **System.Globalization.ThaiBuddhistCalendar** class recognizes only the current era. The **System.DateTime** instance to read.

GetMonth

1
2 [C#] public override int GetMonth(DateTime time);

3 [C++] public: int GetMonth(DateTime time);

4 [VB] Overrides Public Function GetMonth(ByVal time As DateTime) As Integer

5 [JScript] public override function GetMonth(time : DateTime) : int;

6
7 *Description*

8 Gets the month in the specified **System.DateTime** .

9 *Return Value:* An integer between 1 and 12 that represents the month in *time* . The

10 **System.DateTime** instance to read.

11 **GetMonthsInYear**

12
13 [C#] public override int GetMonthsInYear(int year, int era);

14 [C++] public: int GetMonthsInYear(int year, int era);

15 [VB] Overrides Public Function GetMonthsInYear(ByVal year As Integer, ByVal
16 era As Integer) As Integer

17 [JScript] public override function GetMonthsInYear(year : int, era : int) : int; Gets
18 the number of months in the specified year.

19
20 *Description*

21 Gets the number of months in the year specified by the *year* and *era*
22 parameters.

23 *Return Value:* The number of months in the specified year in the specified era. An
24 integer that represents the year. An integer that represents the era.

25 **GetYear**

1
2 [C#] public override int GetYear(DateTime time);

3 [C++] public: int GetYear(DateTime time);

4 [VB] Overrides Public Function GetYear(ByVal time As DateTime) As Integer

5 [JScript] public override function GetYear(time : DateTime) : int;

6
7 *Description*

8 Gets the year in the specified **System.DateTime** .

9 *Return Value:* An integer between 1 and 9999 that represents the year in *time* . The
10 **System.DateTime** instance to read.

11 **IsLeapDay**

12
13 [C#] public override bool IsLeapDay(int year, int month, int day, int era);

14 [C++] public: bool IsLeapDay(int year, int month, int day, int era);

15 [VB] Overrides Public Function IsLeapDay(ByVal year As Integer, ByVal month
16 As Integer, ByVal day As Integer, ByVal era As Integer) As Boolean

17 [JScript] public override function IsLeapDay(year : int, month : int, day : int, era :
18 int) : Boolean; Determines whether the specified day is a leap day.

19
20 *Description*

21 Determines whether the date specified by the *year* , *month* , *day* , and *era*
22 parameters is a leap day.

23 *Return Value:* **true** if the specified day is a leap day; otherwise, **false** .

24 Leap years in the Thai Buddhist calendar correspond to the same leap years
25 in the Gregorian calendar. A common year has 365 days and a leap year has 366

days. An integer that represents the year. An integer that represents the month. An integer that represents the day. An integer that represents the era.

IsLeapMonth

[C#] public override bool IsLeapMonth(int year, int month, int era);

[C++] public: bool IsLeapMonth(int year, int month, int era);

[VB] Overrides Public Function IsLeapMonth(ByVal year As Integer, ByVal month As Integer, ByVal era As Integer) As Boolean

[JScript] public override function IsLeapMonth(year : int, month : int, era : int) : Boolean; Determines whether the specified month is a leap month.

Description

Determines whether the month specified by the *year* , *month* , and *era* parameters is a leap month.

Return Value: This method always returns **false** , unless overridden by a derived class.

Leap years in the Thai Buddhist calendar correspond to the same leap years in the Gregorian calendar. A common year has 365 days and a leap year has 366 days. An integer that represents the year. An integer that represents the month. An integer that represents the era.

IsLeapYear

[C#] public override bool IsLeapYear(int year, int era);

[C++] public: bool IsLeapYear(int year, int era);

[VB] Overrides Public Function IsLeapYear(ByVal year As Integer, ByVal era As

Integer) As Boolean

[JScript] public override function IsLeapYear(year : int, era : int) : Boolean;

Determines whether the specified year is a leap year.

Description

Determines whether the year specified by the *year* and *era* parameters is a leap year.

Return Value: **true** if the specified year is a leap year; otherwise, **false** .

Leap years in the Thai Buddhist calendar correspond to the same leap years in the Gregorian calendar. A common year has 365 days and a leap year has 366 days. An integer that represents the year. An integer that represents the era.

ToDateTime

[C#] public override DateTime ToDateTime(int year, int month, int day, int hour, int minute, int second, int millisecond, int era);

[C++] public: DateTime ToDateTime(int year, int month, int day, int hour, int minute, int second, int millisecond, int era);

[VB] Overrides Public Function ToDateTime(ByVal year As Integer, ByVal month As Integer, ByVal day As Integer, ByVal hour As Integer, ByVal minute As Integer, ByVal second As Integer, ByVal millisecond As Integer, ByVal era As Integer) As DateTime

[JScript] public override function ToDateTime(year : int, month : int, day : int, hour : int, minute : int, second : int, millisecond : int, era : int) : DateTime; Returns a **System.DateTime** that is set to the specified date.

Description

Returns a **System.DateTime** that is set to the specified date and time in the specified era.

Return Value: The **System.DateTime** instance set to the specified date and time in the current era. An integer that represents the year. An integer that represents the month. An integer that represents the day. An integer that represents the hour. An integer that represents the minute. An integer that represents the second. An integer that represents the millisecond. An integer that represents the era.

ToFourDigitYear

[C#] public override int ToFourDigitYear(int year);

[C++] public: int ToFourDigitYear(int year);

[VB] Overrides Public Function ToFourDigitYear(ByVal year As Integer) As Integer

[JScript] public override function ToFourDigitYear(year : int) : int;

Description

Converts the specified two-digit year to a four-digit year by using the **System.Globalization.ThaiBuddhistCalendar.TwoDigitYearMax** property to determine the appropriate century.

Return Value: An integer that contains the four-digit representation of *year*.

System.Globalization.ThaiBuddhistCalendar.TwoDigitYearMax is the last year in the 100-year range that can be represented by a two-digit year. The century is determined by finding the sole occurrence of the two-digit *year* within

that 100-year range. For example, if

System.Globalization.ThaiBuddhistCalendar.TwoDigitYearMax is set to 2029, the 100-year range is from 1930 to 2029; therefore, a 2-digit value of 30 is interpreted as 1930, while a 2-digit value of 29 is interpreted as 2029. A two-digit integer that represents the year to convert.

UnicodeCategory enumeration (System.Globalization)

ToString

Description

Defines the Unicode category of a character.

The Unicode Standard defines the following: A surrogate pair is a coded character representation for a single abstract character that consists of a sequence of two code units, where the first unit of the pair is a high-surrogate and the second is a low-surrogate. A high-surrogate is a Unicode code point in the range U+D800 through U+DBFF and a low-surrogate is a Unicode code point in the range U+DC00 through U+DFFF.

ToString

[C#] public const UnicodeCategory ClosePunctuation;

[C++] public: const UnicodeCategory ClosePunctuation;

[VB] Public Const ClosePunctuation As UnicodeCategory

[JScript] public var ClosePunctuation : UnicodeCategory;

Description

Indicates that the character is the closing character of one of the paired punctuation marks, such as parentheses, square brackets, and braces. The value is 21.

ToString

```
[C#] public const UnicodeCategory ConnectorPunctuation;  
[C++] public: const UnicodeCategory ConnectorPunctuation;  
[VB] Public Const ConnectorPunctuation As UnicodeCategory  
[JScript] public var ConnectorPunctuation : UnicodeCategory;
```

Description

Indicates that the character is a connector punctuation, which connects two characters. The value is 18.

ToString

```
[C#] public const UnicodeCategory Control;  
[C++] public: const UnicodeCategory Control;  
[VB] Public Const Control As UnicodeCategory  
[JScript] public var Control : UnicodeCategory;
```

Description

Indicates that the character is a control code, whose Unicode value is U+007F or in the range U+0000 through U+001F or U+0080 through U+009F. The value is 14.

ToString

[C#] public const UnicodeCategory CurrencySymbol;

[C++] public: const UnicodeCategory CurrencySymbol;

[VB] Public Const CurrencySymbol As UnicodeCategory

[JScript] public var CurrencySymbol : UnicodeCategory;

Description

Indicates that the character is a currency symbol. The value is 26.

ToString

[C#] public const UnicodeCategory DashPunctuation;

[C++] public: const UnicodeCategory DashPunctuation;

[VB] Public Const DashPunctuation As UnicodeCategory

[JScript] public var DashPunctuation : UnicodeCategory;

Description

Indicates that the character is a dash or a hyphen. The value is 19.

ToString

[C#] public const UnicodeCategory DecimalDigitNumber;

[C++] public: const UnicodeCategory DecimalDigitNumber;

[VB] Public Const DecimalDigitNumber As UnicodeCategory

[JScript] public var DecimalDigitNumber : UnicodeCategory;

Description

Indicates that the character is a decimal digit; that is, in the range 0 through 9. The value is 8.

ToString

[C#] public const UnicodeCategory EnclosingMark;

[C++] public: const UnicodeCategory EnclosingMark;

[VB] Public Const EnclosingMark As UnicodeCategory

[JScript] public var EnclosingMark : UnicodeCategory;

Description

Indicates that the character is an enclosing mark, which is a nonspacing combining character that surrounds all previous characters up to and including a base character. The value is 7.

ToString

[C#] public const UnicodeCategory FinalQuotePunctuation;

[C++] public: const UnicodeCategory FinalQuotePunctuation;

[VB] Public Const FinalQuotePunctuation As UnicodeCategory

[JScript] public var FinalQuotePunctuation : UnicodeCategory;

Description

Indicates that the character is a closing or final quotation mark. The value is 23.

ToString

```
[C#] public const UnicodeCategory Format;
[C++] public: const UnicodeCategory Format;
[VB] Public Const Format As UnicodeCategory
[JScript] public var Format : UnicodeCategory;
```

Description

Indicates that the character is a format character, which is not normally rendered but affects the layout of text or the operation of text processes. The value is 15.

ToString

```
[C#] public const UnicodeCategory InitialQuotePunctuation;
[C++] public: const UnicodeCategory InitialQuotePunctuation;
[VB] Public Const InitialQuotePunctuation As UnicodeCategory
[JScript] public var InitialQuotePunctuation : UnicodeCategory;
```

Description

Indicates that the character is an opening or initial quotation mark. The value is 22.

ToString

```
[C#] public const UnicodeCategory LetterNumber;
[C++] public: const UnicodeCategory LetterNumber;
[VB] Public Const LetterNumber As UnicodeCategory
```

1 [JScript] public var LetterNumber : UnicodeCategory;

3 *Description*

4 Indicates that the character is a number represented by a letter, instead of a
5 decimal digit; for example, the Roman numeral for five, which is 'V'. The value is
6 9.

7 ToString

9 [C#] public const UnicodeCategory LineSeparator;

10 [C++] public: const UnicodeCategory LineSeparator;

11 [VB] Public Const LineSeparator As UnicodeCategory

12 [JScript] public var LineSeparator : UnicodeCategory;

14 *Description*

15 Indicates that the character is used to separate lines of text. The value is 12.

16 ToString

18 [C#] public const UnicodeCategory LowercaseLetter;

19 [C++] public: const UnicodeCategory LowercaseLetter;

20 [VB] Public Const LowercaseLetter As UnicodeCategory

21 [JScript] public var LowercaseLetter : UnicodeCategory;

23 *Description*

24 Indicates that the character is a lowercase letter. The value is 1.

25 ToString

1
2 [C#] public const UnicodeCategory MathSymbol;
3 [C++] public: const UnicodeCategory MathSymbol;
4 [VB] Public Const MathSymbol As UnicodeCategory
5 [JScript] public var MathSymbol : UnicodeCategory;
6

7 *Description*

8 Indicates that the character is a mathematical symbol, such as '+' or '='. The
9 value is 25.

10 ToString

11
12 [C#] public const UnicodeCategory ModifierLetter;
13 [C++] public: const UnicodeCategory ModifierLetter;
14 [VB] Public Const ModifierLetter As UnicodeCategory
15 [JScript] public var ModifierLetter : UnicodeCategory;
16

17 *Description*

18 Indicates that the character is a modifier letter, which is free-standing
19 spacing character that indicates modifications of a preceding letter. The value is 3.

20 ToString

21
22 [C#] public const UnicodeCategory ModifierSymbol;
23 [C++] public: const UnicodeCategory ModifierSymbol;
24 [VB] Public Const ModifierSymbol As UnicodeCategory
25 [JScript] public var ModifierSymbol : UnicodeCategory;

1
2 *Description*

3 Indicates that the character is a modifier symbol, which indicates
4 modifications of surrounding characters; for example, the fraction slash indicates
5 that the number to the left is the numerator and the number to the right is the
6 denominator. The value is 27.

7 ToString

8
9 [C#] public const UnicodeCategory NonSpacingMark;

10 [C++] public: const UnicodeCategory NonSpacingMark;

11 [VB] Public Const NonSpacingMark As UnicodeCategory

12 [JScript] public var NonSpacingMark : UnicodeCategory;

13
14 *Description*

15 Indicates that the character is a nonspacing character, which indicates
16 modifications of a base character. The value is 5.

17 ToString

18
19 [C#] public const UnicodeCategory OpenPunctuation;

20 [C++] public: const UnicodeCategory OpenPunctuation;

21 [VB] Public Const OpenPunctuation As UnicodeCategory

22 [JScript] public var OpenPunctuation : UnicodeCategory;

23
24 *Description*

Indicates that the character is the opening character of one of the paired punctuation marks, such as parentheses, square brackets, and braces. The value is 20.

ToString

[C#] public const UnicodeCategory OtherLetter;

[C++] public: const UnicodeCategory OtherLetter;

[VB] Public Const OtherLetter As UnicodeCategory

[JScript] public var OtherLetter : UnicodeCategory;

Description

Indicates that the character is a letter that is not an uppercase letter, a lowercase letter, a titlecase letter, or a modifier letter. The value is 4.

ToString

[C#] public const UnicodeCategory OtherNotAssigned;

[C++] public: const UnicodeCategory OtherNotAssigned;

[VB] Public Const OtherNotAssigned As UnicodeCategory

[JScript] public var OtherNotAssigned : UnicodeCategory;

Description

Indicates that the character is not assigned to any Unicode category. The value is 29.

ToString

```

1
2 [C#] public const UnicodeCategory OtherNumber;
3 [C++] public: const UnicodeCategory OtherNumber;
4 [VB] Public Const OtherNumber As UnicodeCategory
5 [JScript] public var OtherNumber : UnicodeCategory;
6

```

Description

Indicates that the character is a number that is neither a decimal digit nor a letter number; for example, the fraction 1/2. The value is 10.

ToString

```

11
12 [C#] public const UnicodeCategory OtherPunctuation;
13 [C++] public: const UnicodeCategory OtherPunctuation;
14 [VB] Public Const OtherPunctuation As UnicodeCategory
15 [JScript] public var OtherPunctuation : UnicodeCategory;
16

```

Description

Indicates that the character is a punctuation that is not a connector punctuation, a dash punctuation, an open punctuation, a close punctuation, an initial quote punctuation, or a final quote punctuation. The value is 24.

ToString

```

22
23 [C#] public const UnicodeCategory OtherSymbol;
24 [C++] public: const UnicodeCategory OtherSymbol;
25 [VB] Public Const OtherSymbol As UnicodeCategory

```


1 [JScript] public var OtherSymbol : UnicodeCategory;

3 *Description*

4 Indicates that the character is a symbol that is not a mathematical symbol, a
5 currency symbol or a modifier symbol. The value is 28.

6 ToString

8 [C#] public const UnicodeCategory ParagraphSeparator;

9 [C++] public: const UnicodeCategory ParagraphSeparator;

10 [VB] Public Const ParagraphSeparator As UnicodeCategory

11 [JScript] public var ParagraphSeparator : UnicodeCategory;

13 *Description*

14 Indicates that the character is used to separate paragraphs. The value is 13.

15 ToString

17 [C#] public const UnicodeCategory PrivateUse;

18 [C++] public: const UnicodeCategory PrivateUse;

19 [VB] Public Const PrivateUse As UnicodeCategory

20 [JScript] public var PrivateUse : UnicodeCategory;

22 *Description*

23 Indicates that the character is a private-use character, whose Unicode value
24 is in the range U+E000 through U+F8FF. The value is 17.

25 ToString

1
2 [C#] public const UnicodeCategory SpaceSeparator;

3 [C++] public: const UnicodeCategory SpaceSeparator;

4 [VB] Public Const SpaceSeparator As UnicodeCategory

5 [JScript] public var SpaceSeparator : UnicodeCategory;

6
7 *Description*

8 Indicates that the character is a space character, which has no glyph but is
9 not a control or format character. The value is 11.

10 ToString

11
12 [C#] public const UnicodeCategory SpacingCombiningMark;

13 [C++] public: const UnicodeCategory SpacingCombiningMark;

14 [VB] Public Const SpacingCombiningMark As UnicodeCategory

15 [JScript] public var SpacingCombiningMark : UnicodeCategory;

16
17 *Description*

18 Indicates that the character is a spacing character, which indicates
19 modifications of a base character and affects the width of the glyph for that base
20 character. The value is 6.

21 ToString

22
23 [C#] public const UnicodeCategory Surrogate;

24 [C++] public: const UnicodeCategory Surrogate;

25 [VB] Public Const Surrogate As UnicodeCategory

1 [JScript] public var Surrogate : UnicodeCategory;

3 *Description*

4 Indicates that the character is a high-surrogate or a low-surrogate. Surrogate
5 code values are in the range U+D800 through U+DFFF. The value is 16.

6 ToString

8 [C#] public const UnicodeCategory TitlecaseLetter;

9 [C++] public: const UnicodeCategory TitlecaseLetter;

10 [VB] Public Const TitlecaseLetter As UnicodeCategory

11 [JScript] public var TitlecaseLetter : UnicodeCategory;

13 *Description*

14 Indicates that the character is a titlecase letter. The value is 2.

15 ToString

17 [C#] public const UnicodeCategory UppercaseLetter;

18 [C++] public: const UnicodeCategory UppercaseLetter;

19 [VB] Publ

21 SYSTEM.RESOURCES NAMESPACE

22 Creating resources can help developers develop robust, culture-aware
23 programs without having to recompile an application because the resources have
24 changed. Resources are an application-building feature that allows developers to
25 place culture-specific data inside satellite data files (called resource files), rather

1 than directly in a main application. The main assembly does not strong bind to
2 these satellite data files that give developers the flexibility to deploy them in
3 different phases. When building an application, the developer can identify aspects
4 that are culture-specific such as user visible strings, graphics etc., and put these in
5 a different resources file for each culture where the application may be used. At
6 run time, the appropriate set of resources will be loaded, based on the user's
7 culture settings. The specific setting used is the **CurrentUICulture** for the main
8 thread of execution, which the user can set programmatically.

9 The ResourceManager class provides the user with the ability to access and
10 control resources stored in the main assembly or in resource satellite assemblies.
11 Use the ResourceManager.GetObject and ResourceManager.GetString methods to
12 retrieve culture-specific objects and strings, as illustrated in the following
13 example.

```
14 class Class1
15 {
16     //Creates a resource manger bound to the localizable
17     //resource file associated with this assembly.
18     static ResourceManager rm = new ResourceManager(
19         "strings",Assembly.GetExecutingAssembly());
20
21     static void Main(string[] args)
22     {
23         //Pulls the string with the key "Hello" out of the
24         //resource file that is the best match for the current
25         //culture.
26         Console.WriteLine (rm.GetString ("Hello"));
27     }
28 }
```

22 SYSTEM.NET NAMESPACE

23 The System.Net namespace provides a simple programming interface to
24 many of the protocols found on the network today. The WebRequest and
25 WebResponse classes form the basis of "pluggable protocols," an implementation

1 of network services that enables developers to develop applications that use
2 Internet resources without worrying about the specific details of the protocol used.

3 The System.Net.Sockets namespace provides a managed implementation of
4 the Windows Sockets interface for developers that need to tightly control access to
5 the network. Developers familiar with the Winsock API can readily develop
6 applications using the Socket class.

7 The TCPClient, TCPListener, and UDPClient classes encapsulate the
8 details of creating TCP and UDP connections to the Internet.

9 The following is a more detailed description of the System.Net namespace,
10 identifying various classes, interfaces, enumerations, and so forth contained in the
11 System.Net and System.Net.Sockets namespaces.

12 System.Net

13 The namespace provides a simple programming interface to many of the
14 protocols found on the network today. The and classes form the basis of
15 "pluggable protocols," an implementation of network services that enables you to
16 develop applications that use Internet resources without worrying about the
17 specific details of the protocol used.

18 *Description*

19 The **System.Net** namespace provides a simple programming interface to
20 many of the protocols found on the network today. The **System.Net.WebRequest**
21 and **System.Net.WebResponse** classes form the basis of "pluggable protocols," an
22 implementation of network services that enables you to develop applications that
23 use Internet resources without worrying about the specific details of the protocol
24 used.
25

AuthenticationManager class (System.Net)

Description

Manages the authentication modules called during the client authentication process.

System.Net.AuthenticationManager is a static class that manages the authentication modules that an application uses. When a request is made to protected resources, the **System.Net.AuthenticationManager** calls the **System.Net.AuthenticationManager.Authenticate(System.String,System.Net.WebRequest,System.Net.ICredentials)** method to get an **System.Net.Authorization** instance to use in subsequent requests.

Properties:

RegisteredModules

[C#] public static IEnumerable RegisteredModules {get;}

[C++] public: __property static IEnumerable* get_RegisteredModules();

[VB] Public Shared ReadOnly Property RegisteredModules As IEnumerable

[JScript] public static function get RegisteredModules() : IEnumerable;

Description

Gets a list of authentication modules that are registered with the authentication manager.

The **System.Net.AuthenticationManager.RegisteredModules** property provides an **System.Collections.IEnumerator** instance that enables the list of

registered authentication modules to be read. The

System.Net.AuthenticationManager.Register(System.Net.IAuthenticationModule) method adds modules to the list, and the

System.Net.AuthenticationManager.Unregister(System.Net.IAuthenticationModule) method removes modules from it.

Methods:

Authenticate

[C#] public static Authorization Authenticate(string challenge, WebRequest request, ICredentials credentials);

[C++] public: static Authorization* Authenticate(String* challenge, WebRequest* request, ICredentials* credentials);

[VB] Public Shared Function Authenticate(ByVal challenge As String, ByVal request As WebRequest, ByVal credentials As ICredentials) As Authorization

[JScript] public static function Authenticate(challenge : String, request : WebRequest, credentials : ICredentials) : Authorization;

Description

Calls each registered authentication module to find the first module that can respond to the authentication request.

Return Value: An instance of the **System.Net.Authorization** class containing the result of the authorization attempt. If there is no authentication module to respond to the challenge, this method returns **null**.

The

System.Net.AuthenticationManager.Authenticate(System.String, System.Net.

WebRequest, System.Net.ICredentials) method calls the **System.Net.IAuthenticationModule.Authenticate(System.String, System.Net.WebRequest, System.Net.ICredentials)** method on each registered authentication module until one of the module responds with an **System.Net.Authorization** instance. The challenge returned by the Internet resource. The **System.Net.WebRequest** that initiated the authentication challenge. The **System.Net.ICredentials** associated with this request.

PreAuthenticate

[C#] public static Authorization PreAuthenticate(WebRequest request, ICredentials credentials);
[C++] public: static Authorization* PreAuthenticate(WebRequest* request, ICredentials* credentials);
[VB] Public Shared Function PreAuthenticate(ByVal request As WebRequest, ByVal credentials As ICredentials) As Authorization
[JScript] public static function PreAuthenticate(request : WebRequest, credentials : ICredentials) : Authorization;

Description

Preauthenticates a request.

Return Value: An instance of the **System.Net.Authorization** class if the request can be preauthenticated; otherwise, **null** . If *credentials* is **null** , this method returns **null** .

If the authentication module can preauthenticate the request, the **PreAuthenticate** method returns an **Authentication** instance and sends the

1 authorization information to the server preemptively instead of waiting for the
2 resource to issue a challenge. This behavior is outlined in section 3.3 of RFC 2617
3 (HTTP Authentication: Basic and Digest Access Authentication). Authentication
4 modules that support preauthentication allow clients to improve server efficiency
5 by avoiding extra round trips caused by authentication challenges. A
6 **System.Net.WebRequest** to an Internet resource. The **System.Net.ICredentials**
7 associated with the request.

8 Register

9
10 [C#] public static void Register(IAAuthenticationModule authenticationModule);

11 [C++] public: static void Register(IAAuthenticationModule*
12 authenticationModule);

13 [VB] Public Shared Sub Register(ByVal authenticationModule As
14 IAAuthenticationModule)

15 [JScript] public static function Register(authenticationModule :
16 IAAuthenticationModule);

17 Description

18 Registers an authentication module with the authentication manager.

19 The

20
21 **System.Net.AuthenticationManager.Register(System.Net.IAuthenticationMo
22 dule)** method adds authentication modules to the end of the list of modules called
23 by the

24 **System.Net.AuthenticationManager.Authenticate(System.String,System.Net.
25 WebRequest,System.Net.ICredentials)** method. Authentication modules are

called in the order in which they were added to the list. The

System.Net.IAuthenticationModule to register with the authentication manager.

Unregister

[C#] public static void Unregister(IAuthenticationModule authenticationModule);

[C++] public: static void Unregister(IAuthenticationModule*

authenticationModule);

[VB] Public Shared Sub Unregister(ByVal authenticationModule As

IAuthenticationModule)

[JScript] public static function Unregister(authenticationModule :

IAuthenticationModule); Removes authentication modules from the list of

registered modules.

Description

Removes the specified authentication module from the list of registered modules.

The

System.Net.AuthenticationManager.Unregister(System.Net.IAuthentication

Module) method removes the specified authentication module from the list of

authentication modules called by the

System.Net.AuthenticationManager.Authenticate(System.String, System.Net.

WebRequest, System.Net.ICredentials) method. The module must have been

added to the list using the

System.Net.AuthenticationManager.Register(System.Net.IAuthenticationMo

dule) method before it can be removed from the list. The **IAuthentication** module to remove.

Unregister

[C#] public static void Unregister(string authenticationScheme);

[C++] public: static void Unregister(String* authenticationScheme);

[VB] Public Shared Sub Unregister(ByVal authenticationScheme As String)

[JScript] public static function Unregister(authenticationScheme : String);

Description

Removes authentication modules with the specified authentication scheme from the list of registered modules.

The

System.Net.AuthenticationManager.Unregister(System.Net.IAuthentication Module) method removes the authentication module with the specified

authentication scheme from the list of authentication modules called by the

System.Net.AuthenticationManager.Authenticate(System.String, System.Net.WebRequest, System.Net.ICredentials) method. The module must have been

added to the list using the

System.Net.AuthenticationManager.Register(System.Net.IAuthenticationMo

dule) method before it can be removed from the list. The authentication scheme of the module to remove.

Authorization class (System.Net)

Unregister

1
2
3 *Description*

4 Contains an authentication message for an Internet server.

5 The **System.Net.AuthenticationManager** returns an instance of the
6 **System.Net.Authorization** class containing the authentication message that is
7 sent to the Internet server to indicate that the client (such as
8 **System.Net.WebRequest** or one of its descendants) is authorized to access the
9 server.

10 Constructors:

11 Authorization

12 *Example Syntax:*

13 Unregister

14
15 [C#] public Authorization(string token);

16 [C++] public: Authorization(String* token);

17 [VB] Public Sub New(ByVal token As String)

18 [JScript] public function Authorization(token : String); Creates a new instance of
19 the **System.Net.Authorization** class.

20
21 *Description*

22 Creates a new instance of the **System.Net.Authorization** class with the
23 specified authorization message.

24 The **System.Net.Authorization** instance is created with the
25 **System.Net.Authorization.Message** property set to *token* and the

System.Net.Authorization.Complete property set to **true** . The encrypted authorization message expected by the server.

Authorization

Example Syntax:

Unregister

[C#] public Authorization(string token, bool finished);

[C++] public: Authorization(String* token, bool finished);

[VB] Public Sub New(ByVal token As String, ByVal finished As Boolean)

[JScript] public function Authorization(token : String, finished : Boolean);

Description

Creates a new instance of the **System.Net.Authorization** class with the specified authorization message and completion status.

The **System.Net.Authorization** instance is created with the **System.Net.Authorization.Message** property set to *token* and the **System.Net.Authorization.Complete** property set to *finished* . The encrypted authorization message expected by the server . The completion status of the authorization attempt.

Authorization

Example Syntax:

Unregister

[C#] public Authorization(string token, bool finished, string connectionGroupId);

[C++] public: Authorization(String* token, bool finished, String*

1 connectionGroupId);

2 [VB] Public Sub New(ByVal token As String, ByVal finished As Boolean, ByVal
3 connectionGroupId As String)

4 [JScript] public function Authorization(token : String, finished : Boolean,
5 connectionGroupId : String);

6 7 *Description*

8 Creates a new instance of the **System.Net.Authorization** class with the
9 specified authorization message, completion status, and connection group
10 identifier. The encrypted authorization message expected by the server . The
11 completion status of the authorization attempt. A unique identifier that can be used
12 to create private Client-Server connections, that would only be bound to this
13 authentication scheme.

14 Complete

15 Unregister

16
17 [C#] public bool Complete {get;}

18 [C++] public: __property bool get_Complete();

19 [VB] Public ReadOnly Property Complete As Boolean

20 [JScript] public function get Complete() : Boolean;

21 22 *Description*

23 Gets the completion status of the authorization.

24 The **System.Net.Authorization.Complete** property is set to **true** when the
25 authentication process between the client and the server is finished. Some

1 authentication modules, such as the Kerberos module, use multiple round trips
2 between the client and server to complete the authentication process. To keep the
3 **System.Net.WebRequest** or descendant that initiated the authentication process
4 from interrupting while authorization is taking place, the authentication module
5 sets the **System.Net.Authorization.Complete** property to **false** .

6 ConnectionGroupId

7 Unregister

9 [C#] public string ConnectionGroupId {get;}

10 [C++] public: __property String* get_ConnectionGroupId();

11 [VB] Public ReadOnly Property ConnectionGroupId As String

12 [JScript] public function get ConnectionGroupId() : String;

14 *Description*

15 Gets a unique identifier for user-specific connections.

16 The **System.Net.Authorization.ConnectionGroupId** property is a unique
17 string that associates a connection with a specific authenticating entity. For
18 example, the NTLM authorization module ties the authentication credential
19 information to a specific connection to prevent invalid reuse of the connection.

20 Message

21 Unregister

23 [C#] public string Message {get;}

24 [C++] public: __property String* get_Message();

25 [VB] Public ReadOnly Property Message As String

1 [JScript] public function get Message() : String;

2
3 *Description*

4 Gets the message returned to the server in response to an authentication
5 challenge.

6 The **System.Net.Authorization.Message** property contains the
7 authorization string that the client will return to the server when accessing
8 protected resources. The actual contents of the message is defined by the
9 authentication type the client and server are using. Basic HTTP authentication, for
10 example, uses a different message than Kerberos authentication.

11 ProtectionRealm

12 Unregister

13
14 [C#] public string[] ProtectionRealm {get; set;}

15 [C++] public: __property String* get_ProtectionRealm();public: __property void
16 set_ProtectionRealm(String* __gc[]);

17 [VB] Public Property ProtectionRealm As String ()

18 [JScript] public function get ProtectionRealm() : String[];public function set
19 ProtectionRealm(String[]);

20
21 *Description*

22 Gets or sets the prefix for uniform resource identifiers (URIs) that can be
23 authenticated with the **System.Net.Authorization.Message** property.

1 The **System.Net.Authorization.ProtectionRealm** property contains a list
2 of URI prefixes that the **System.Net.Authorization.Message** property can be used
3 to authenticate.

4 Cookie class (System.Net)

5 ToString

6
7
8 *Description*

9 Provides a set of properties and methods used to manage cookies. This
10 class cannot be inherited.

11 For a list of initial property values for an instance of **System.Net.Cookie** ,
12 see the **System.Net.Cookie.#ctor** constructors.

13 Cookie

14 *Example Syntax:*

15 ToString

16
17 [C#] public Cookie();

18 [C++] public: Cookie();

19 [VB] Public Sub New()

20 [JScript] public function Cookie(); Initializes a new instance of the

21 **System.Net.Cookie** class.

22
23 *Description*

24 Initializes a new instance of the **System.Net.Cookie** class using the empty
25 string for default parameters: *name, value, path, domain*.

Cookie

Example Syntax:

ToString

[C#] public Cookie(string name, string value);

[C++] public: Cookie(String* name, String* value);

[VB] Public Sub New(ByVal name As String, ByVal value As String)

[JScript] public function Cookie(name : String, value : String);

Description

Initializes a new instance of the **System.Net.Cookie** class with specified name and value, using the empty string for default parameters: *path*, *domain*.

string string

Cookie

Example Syntax:

ToString

[C#] public Cookie(string name, string value, string path);

[C++] public: Cookie(String* name, String* value, String* path);

[VB] Public Sub New(ByVal name As String, ByVal value As String, ByVal path As String)

[JScript] public function Cookie(name : String, value : String, path : String);

Description

1 Initializes a new instance of the **System.Net.Cookie** class with specified
2 name, value, and path, using the empty string for default parameter *domain*. string
3 string string

4 Cookie

5 *Example Syntax:*

6 ToString

7
8 [C#] public Cookie(string name, string value, string path, string domain);

9 [C++] public: Cookie(String* name, String* value, String* path, String* domain);

10 [VB] Public Sub New(ByVal name As String, ByVal value As String, ByVal path
11 As String, ByVal domain As String)

12 [JScript] public function Cookie(name : String, value : String, path : String,
13 domain : String);

14
15 *Description*

16 Initializes a new instance of the **System.Net.Cookie** class with specified
17 name, value, path, and domain. string string string string

18 Comment

19 ToString

20
21 [C#] public string Comment {get; set;}

22 [C++] public: __property String* get_Comment();public: __property void
23 set_Comment(String*);

24 [VB] Public Property Comment As String

25 [JScript] public function get Comment() : String;public function set

1 Comment(String);

3 *Description*

4 Gets a comment that the server can add to the cookie.

5 The client can inspect this optional comment for information added by the
6 server about issues such as the privacy policy and so on.

7 CommentUri

8 ToString

10 [C#] public Uri CommentUri {get; set;}

11 [C++] public: __property Uri* get_CommentUri();public: __property void
12 set_CommentUri(Uri*);

13 [VB] Public Property CommentUri As Uri

14 [JScript] public function get CommentUri() : Uri;public function set
15 CommentUri(Uri);

17 *Description*

18 Gets a URI that the server can provide with a cookie.

19 The URI can provide optional information such as how the server uses the
20 cookie.

21 Discard

22 ToString

24 [C#] public bool Discard {get; set;}

25 [C++] public: __property bool get_Discard();public: __property void

1 set_Discard(bool);

2 [VB] Public Property Discard As Boolean

3 [JScript] public function get Discard() : Boolean;public function set

4 Discard(Boolean);

5
6 *Description*

7 Gets the discard flag set by the server.

8 When true, this property instructs the Web browser not to save the cookie
9 on the user's hard drive when a session ends.

10 Domain

11 ToString

12
13 [C#] public string Domain {get; set;}

14 [C++] public: __property String* get_Domain();public: __property void

15 set_Domain(String*);

16 [VB] Public Property Domain As String

17 [JScript] public function get Domain() : String;public function set Domain(String);

18
19 *Description*

20 Gets the URI for which the cookie is valid.

21 A server cannot indicate a domain other than its own. However it can
22 indicate more than one server.

23 Expired

24 ToString

[C#] public bool Expired {get; set;}

[C++] public: __property bool get_Expired();public: __property void
set_Expired(bool);

[VB] Public Property Expired As Boolean

[JScript] public function get Expired() : Boolean;public function set
Expired(Boolean);

Description

Gets the current state of the cookie.

Expires

ToString

[C#] public DateTime Expires {get; set;}

[C++] public: __property DateTime get_Expires();public: __property void
set_Expires(DateTime);

[VB] Public Property Expires As DateTime

[JScript] public function get Expires() : DateTime;public function set
Expires(DateTime);

Description

Gets the expiration **DateTime** for the cookie.

Name

ToString

1
2 [C#] public string Name {get; set;}

3 [C++] public: __property String* get_Name();public: __property void

4 set_Name(String*);

5 [VB] Public Property Name As String

6 [JScript] public function get Name() : String;public function set Name(String);

7
8 *Description*

9 Gets the name for the cookie.

10 Path

11 ToString

12
13 [C#] public string Path {get; set;}

14 [C++] public: __property String* get_Path();public: __property void

15 set_Path(String*);

16 [VB] Public Property Path As String

17 [JScript] public function get Path() : String;public function set Path(String);

18
19 *Description*

20 Gets the URLs to which the cookie applies on the server .

21 Port

22 ToString

23
24 [C#] public string Port {get; set;}

25 [C++] public: __property String* get_Port();public: __property void

1 set_Port(String*);

2 [VB] Public Property Port As String

3 [JScript] public function get Port() : String;public function set Port(String);

4
5 *Description*

6 Gets a list of TCP ports to which the cookie applies.

7 Secure

8 ToString

9
10 [C#] public bool Secure {get; set;}

11 [C++] public: __property bool get _Secure();public: __property void

12 set _Secure(bool);

13 [VB] Public Property Secure As Boolean

14 [JScript] public function get Secure() : Boolean;public function set

15 Secure(Boolean);

16
17 *Description*

18 Gets the security level set by the server.

19 TimeStamp

20 ToString

21
22 [C#] public DateTime TimeStamp {get;}

23 [C++] public: __property DateTime get _TimeStamp();

24 [VB] Public ReadOnly Property TimeStamp As DateTime

25 [JScript] public function get TimeStamp() : DateTime;

1
2 *Description*

3 Gets the **DateTime** when the cookie was issued.

4 Value

5 ToString

6
7 [C#] public string Value {get; set;}

8 [C++] public: __property String* get_Value();public: __property void

9 set_Value(String*);

10 [VB] Public Property Value As String

11 [JScript] public function get Value() : String;public function set Value(String);

12
13 *Description*

14 Gets the server-supplied value for the cookie.

15 Version

16 ToString

17
18 [C#] public int Version {get; set;}

19 [C++] public: __property int get_Version();public: __property void

20 set_Version(int);

21 [VB] Public Property Version As Integer

22 [JScript] public function get Version() : int;public function set Version(int);

23
24 *Description*

25 Gets the version of HTTP state maintenance to which the cookie conforms.

Equals

[C#] public override bool Equals(object comparand);

[C++] public: bool Equals(Object* comparand);

[VB] Overrides Public Function Equals(ByVal comparand As Object) As Boolean

[JScript] public override function Equals(comparand : Object) : Boolean;

Description

An override of the **Object.Equals** method. A reference to a **Cookie** object.

GetHashCode

[C#] public override int GetHashCode();

[C++] public: int GetHashCode();

[VB] Overrides Public Function GetHashCode() As Integer

[JScript] public override function GetHashCode() : int;

Description

An override of **Object.GetHashCode** .

ToString

[C#] public override string ToString();

[C++] public: String* ToString();

[VB] Overrides Public Function ToString() As String

[JScript] public override function ToString() : String;

1
2 *Description*

3 An override of **Object.ToString** .

4 **CookieCollection** class (System.Net)

5 **ToString**

6
7
8 *Description*

9 Provides a collection container for **Cookie** or **CookieCollection** instances.

10 The **CookieCollection** class implements an **ICollection** interface to
11 provide a general mechanism for handling collections of cookies. For example,
12 this is useful in the case where an application wants to act on behalf of multiple
13 users and store cookies for each user.

14 **CookieCollection**

15 *Example Syntax:*

16 **ToString**

17
18 [C#] public **CookieCollection**();

19 [C++] public: **CookieCollection**();

20 [VB] Public Sub New()

21 [JScript] public function **CookieCollection**();

22
23 *Description*

24 Initializes a new instance of the **System.Net.CookieCollection** class.

25 **Count**

ToString

[C#] public int Count {get;}

[C++] public: __property int get_Count();

[VB] Public ReadOnly Property Count As Integer

[JScript] public function get Count() : int;

Description

Gets the number of elements contained in the **CookieCollection** .

IsReadOnly

ToString

[C#] public bool IsReadOnly {get;}

[C++] public: __property bool get_IsReadOnly();

[VB] Public ReadOnly Property IsReadOnly As Boolean

[JScript] public function get IsReadOnly() : Boolean;

Description

Gets or sets a value indicating whether the **CookieCollection** instance is read-only.

IsSynchronized

ToString

[C#] public bool IsSynchronized {get;}

[C++] public: __property bool get_IsSynchronized();

1 [VB] Public ReadOnly Property IsSynchronized As Boolean

2 [JScript] public function get IsSynchronized() : Boolean;

3
4 *Description*

5 Gets a value that indicates whether access to a **CookieCollection** is thread-
6 safe.

7 **SyncRoot** returns an object that can be used to synchronize access to the
8 **CookieCollection** .

9 Item

10 ToString

11
12 [C#] public Cookie this[int index] {get;}

13 [C++] public: __property Cookie* get_Item(int index);

14 [VB] Public Default ReadOnly Property Item(ByVal index As Integer) As Cookie

15 [JScript] returnValue = CookieCollectionObject.Item(index); Gets a specific

16 **CookieCollection** element.

17
18 *Description*

19 Gets the **CookieCollection** element with a specific index. The zero-based
20 index of the **Cookie**.

21 Item

22 ToString

23
24 [C#] public Cookie this[string name] {get;}

25 [C++] public: __property Cookie* get_Item(String* name);

1 [VB] Public Default ReadOnly Property Item(ByVal name As String) As Cookie

2 [JScript] returnValue = CookieCollectionObject.Item(name);

3
4 *Description*

5 Gets the **CookieCollection** element with a specific name. The name of the

6 **Cookie.**

7 SyncRoot

8 ToString

9
10 [C#] public object SyncRoot {get;}

11 [C++] public: __property Object* get_SyncRoot();

12 [VB] Public ReadOnly Property SyncRoot As Object

13 [JScript] public function get SyncRoot() : Object;

14
15 *Description*

16 Gets an object that you can use to synchronize access to the

17 **CookieCollection .**

18 Add

19
20 [C#] public void Add(Cookie cookie);

21 [C++] public: void Add(Cookie* cookie);

22 [VB] Public Sub Add(ByVal cookie As Cookie)

23 [JScript] public function Add(cookie : Cookie); Adds an item to the

24 **CookieCollection .**

1
2 *Description*

3 Adds a **Cookie** to the **CookieCollection** . The **Cookie** to be added to the
4 collection

5 Add

6
7 [C#] public void Add(CookieCollection cookies);

8 [C++] public: void Add(CookieCollection* cookies);

9 [VB] Public Sub Add(ByVal cookies As CookieCollection)

10 [JScript] public function Add(cookies : CookieCollection);

11
12 *Description*

13 Adds a **CookieCollection** to the **CookieCollection** . The **CookieCollection**
14 to be added to the collection

15 CopyTo

16
17 [C#] public void CopyTo(Array array, int index);

18 [C++] public: __sealed void CopyTo(Array* array, int index);

19 [VB] NotOverridable Public Sub CopyTo(ByVal array As Array, ByVal index As
20 Integer)

21 [JScript] public function CopyTo(array : Array, index : int);

22
23 *Description*

24 Copies the elements of the collection to an **Array** , starting at a particular
25 index.

The **Array** must be a one-dimensional array with zero-based indexing. The target **Array**. The zero-based index in *array* at which copying begins.

GetEnumerator

[C#] public IEnumerator GetEnumerator();

[C++] public: __sealed IEnumerator* GetEnumerator();

[VB] NotOverridable Public Function GetEnumerator() As IEnumerator

[JScript] public function GetEnumerator() : IEnumerator;

Description

Gets an enumerator that you can use to iterate through a **CookieCollection**

Return Value: An **IEnumerator** that you can use to iterate through a **CookieCollection**.

You should only use Enumerators to read data in the collection. Enumerators cannot be used to modify the underlying collection. The enumerator does not have exclusive access to the collection.

CookieContainer class (System.Net)

ToString

Description

Contains **CookieCollection** objects.

ToString


```

1
2 [C#] public const int DefaultCookieLengthLimit;
3 [C++] public: const int DefaultCookieLengthLimit;
4 [VB] Public Const DefaultCookieLengthLimit As Integer
5 [JScript] public var DefaultCookieLengthLimit : int;
6

```

Description

Gets the maximum size, in bytes, of elements the **CookieContainer** can hold. This field is constant.

ToString

```

11
12 [C#] public const int DefaultCookieLimit;
13 [C++] public: const int DefaultCookieLimit;
14 [VB] Public Const DefaultCookieLimit As Integer
15 [JScript] public var DefaultCookieLimit : int;
16

```

Description

Gets the maximum number of elements the **CookieContainer** can hold. This field is constant.

ToString

```

21
22 [C#] public const int DefaultPerDomainCookieLimit;
23 [C++] public: const int DefaultPerDomainCookieLimit;
24 [VB] Public Const DefaultPerDomainCookieLimit As Integer
25 [JScript] public var DefaultPerDomainCookieLimit : int;

```

1
2 *Description*

3 Gets the maximum number of elements the **CookieContainer** can
4 reference per domain. This field is constant.

5 **CookieContainer**

6 *Example Syntax:*

7 **ToString**

8
9 [C#] public CookieContainer();

10 [C++] public: CookieContainer();

11 [VB] Public Sub New()

12 [JScript] public function CookieContainer(); Initializes a new instance of the
13 **System.Net.CookieContainer** class.

14
15 *Description*

16 Initializes a new instance of the **System.Net.CookieContainer** class with
17 default values for **DefaultCookieLimit** , **DefaultPerDomainCookieLimit** , and
18 **DefaultCookieLengthLimit** .

19 **CookieContainer**

20 *Example Syntax:*

21 **ToString**

22
23 [C#] public CookieContainer(int capacity);

24 [C++] public: CookieContainer(int capacity);

25 [VB] Public Sub New(ByVal capacity As Integer)

1 [JScript] public function CookieContainer(capacity : int);

3 *Description*

4 Initializes a new instance of the **System.Net.CookieContainer** class with a
5 specified value for the number of elements the container can hold and default
6 values for **DefaultPerDomainCookieLimit** , and **DefaultCookieLengthLimit** .

7 The number of cookies the **CookieContainer** can hold.

8 CookieContainer

9 *Example Syntax:*

10 ToString

12 [C#] public CookieContainer(int capacity, int perDomainCapacity, int
13 maxCookieSize);

14 [C++] public: CookieContainer(int capacity, int perDomainCapacity, int
15 maxCookieSize);

16 [VB] Public Sub New(ByVal capacity As Integer, ByVal perDomainCapacity As
17 Integer, ByVal maxCookieSize As Integer)

18 [JScript] public function CookieContainer(capacity : int, perDomainCapacity : int,
19 maxCookieSize : int);

21 *Description*

22 Initializes a new instance of the **System.Net.CookieContainer** class with a
23 specified value for the number of elements the container can hold, the number of
24 cookies per domain, and the maximum element length . The number of elements

the **CookieContainer** can hold. The number of elements per domain. The maximum size of the elements **CookieContainer** can hold.

Capacity

ToString

[C#] public int Capacity {get; set;}

[C++] public: __property int get_Capacity();public: __property void set_Capacity(int);

[VB] Public Property Capacity As Integer

[JScript] public function get Capacity() : int;public function set Capacity(int);

Description

Gets the number of elements the **CookieContainer** can hold.

Count

ToString

[C#] public int Count {get;}

[C++] public: __property int get_Count();

[VB] Public ReadOnly Property Count As Integer

[JScript] public function get Count() : int;

Description

Gets the number of elements the **CookieContainer** currently holds.

MaxCookieSize

ToString

1
2 [C#] public int MaxCookieSize {get; set;}

3 [C++] public: __property int get_MaxCookieSize();public: __property void
4 set_MaxCookieSize(int);

5 [VB] Public Property MaxCookieSize As Integer

6 [JScript] public function get MaxCookieSize() : int;public function set
7 MaxCookieSize(int);

8
9 *Description*

10 Gets the maximum size of the elements held by the **CookieContainer** .

11 PerDomainCapacity

12 ToString

13
14 [C#] public int PerDomainCapacity {get; set;}

15 [C++] public: __property int get_PerDomainCapacity();public: __property void
16 set_PerDomainCapacity(int);

17 [VB] Public Property PerDomainCapacity As Integer

18 [JScript] public function get PerDomainCapacity() : int;public function set
19 PerDomainCapacity(int);

20
21 *Description*

22 Gets the number of elements allowed per domain.

23 Add

24
25 [C#] public void Add(Cookie cookie);

1 [C++] public: void Add(Cookie* cookie);

2 [VB] Public Sub Add(ByVal cookie As Cookie)

3 [JScript] public function Add(cookie : Cookie); Adds elements to the

4 **CookieContainer** .

5
6 *Description*

7 Adds a **Cookie** to the **CookieContainer** . The **Cookie** to be added to the

8 **CookieContainer** .

9 Add

10
11 [C#] public void Add(CookieCollection cookies);

12 [C++] public: void Add(CookieCollection* cookies);

13 [VB] Public Sub Add(ByVal cookies As CookieCollection)

14 [JScript] public function Add(cookies : CookieCollection);

15
16 *Description*

17 Adds a **CookieCollection** to the **CookieContainer** . The **CookieCollection**
18 to be added to the **CookieContainer** .

19 Add

20
21 [C#] public void Add(Uri uri, Cookie cookie);

22 [C++] public: void Add(Uri* uri, Cookie* cookie);

23 [VB] Public Sub Add(ByVal uri As Uri, ByVal cookie As Cookie)

24 [JScript] public function Add(uri : Uri, cookie : Cookie);

1
2 *Description*

3 Adds a **Cookie** to the **CookieContainer** . The URI of the **Cookie** to be
4 added to the **CookieContainer** . The **Cookie** to be added to the **CookieContainer**
5 .

6 **Add**

7
8 [C#] public void Add(Uri uri, CookieCollection cookies);

9 [C++] public: void Add(Uri* uri, CookieCollection* cookies);

10 [VB] Public Sub Add(ByVal uri As Uri, ByVal cookies As CookieCollection)

11 [JScript] public function Add(uri : Uri, cookies : CookieCollection);

12
13 *Description*

14 Adds a **CookieCollection** to the **CookieContainer** . The URI of the
15 **CookieCollection** to be added to the **CookieContainer** . The **CookieCollection** to
16 be added to the **CookieContainer** .

17 **GetCookieHeader**

18
19 [C#] public string GetCookieHeader(Uri uri);

20 [C++] public: String* GetCookieHeader(Uri* uri);

21 [VB] Public Function GetCookieHeader(ByVal uri As Uri) As String

22 [JScript] public function GetCookieHeader(uri : Uri) : String;

23
24 *Description*

Gets the HTTP header of the element associated with a specific URI. The URI desired.

GetCookies

[C#] public CookieCollection GetCookies(Uri uri);

[C++] public: CookieCollection* GetCookies(Uri* uri);

[VB] Public Function GetCookies(ByVal uri As Uri) As CookieCollection

[JScript] public function GetCookies(uri : Uri) : CookieCollection;

Description

Gets the elements associated with a specific URI.

Return Value: A **CookieCollection** containing the elements associated with a specific URI. The URI of the elements desired.

SetCookies

[C#] public void SetCookies(Uri uri, string cookieHeader);

[C++] public: void SetCookies(Uri* uri, String* cookieHeader);

[VB] Public Sub SetCookies(ByVal uri As Uri, ByVal cookieHeader As String)

[JScript] public function SetCookies(uri : Uri, cookieHeader : String);

Description

Associates a cookie header with a specific URI . The URI of the header.

CookieException class (System.Net)

ToString

1
2
3 *Description*

4 The exception that is thrown when XXX.

5 CookieException

6 *Example Syntax:*

7 ToString

8
9 [C#] public CookieException();

10 [C++] public: CookieException();

11 [VB] Public Sub New()

12 [JScript] public function CookieException(); Initializes a new instance of the

13 **System.Net.CookieException** class.

14
15 *Description*

16 Initializes a new instance of the **System.Net.CookieException** class using
17 default parameters.

18 CookieException

19 *Example Syntax:*

20 ToString

21
22 [C#] protected CookieException(SerializationInfo serializationInfo,

23 StreamingContext streamingContext);

24 [C++] protected: CookieException(SerializationInfo* serializationInfo,

25 StreamingContext streamingContext);

[VB] Protected Sub New(ByVal serializationInfo As SerializationInfo, ByVal
streamingContext As StreamingContext)
[JScript] protected function CookieException(serializationInfo : SerializationInfo,
streamingContext : StreamingContext);

Description

Initializes a new instance of the **System.Net.CookieException** class with
specified values of *serializationInfo* and *streamingContext* . The **SerializationInfo**
to be used. The **StreamingContext** to be used.

HelpLink

HResult

InnerException

Message

Source

StackTrace

TargetSite

ISerializable.GetObjectData

[C#] void ISerializable.GetObjectData(SerializationInfo serializationInfo,
StreamingContext streamingContext);

[C++] void ISerializable::GetObjectData(SerializationInfo* serializationInfo,
StreamingContext streamingContext);

[VB] Sub GetObjectData(ByVal serializationInfo As SerializationInfo, ByVal
streamingContext As StreamingContext) Implements ISerializable.GetObjectData

1 [JScript] function ISerializable.GetObjectData(serializationInfo : SerializationInfo,
2 streamingContext : StreamingContext);

3 CredentialCache class (System.Net)

4 ToString

5
6
7 *Description*

8 Provides storage for multiple credentials.

9 The **System.Net.CredentialCache** class stores credentials for multiple
10 Internet resources. Applications that need to access multiple resources can store
11 the credentials for those resources in a **System.Net.CredentialCache** instance that
12 then provides the proper set of credentials to the Internet resource when required.

13 When the

14 **System.Net.CredentialCache.GetCredential(System.Uri,System.String)**
15 method is called, it compares the URI and authentication type provided with those
16 stored in the cache and returns the first set of credentials that match.

17 CredentialCache

18 *Example Syntax:*

19 ToString

20
21 [C#] public CredentialCache();

22 [C++] public: CredentialCache();

23 [VB] Public Sub New()

24 [JScript] public function CredentialCache();

Description

Creates a new instance of the **System.Net.CredentialCache** class.

The constructor creates a **System.Net.CredentialCache** instance with its **System.Net.CredentialCache.DefaultCredentials** property initialized to the system credentials of the current security context. For client applications, these represent the user name, password, and domain of the user who is currently logged in. For ASP.NET applications, the credential is the process token of the IIS server or the process token being impersonated by the IIS server.

DefaultCredentials

ToString

[C#] public static ICredentials DefaultCredentials {get;}

[C++] public: __property static ICredentials* get_DefaultCredentials();

[VB] Public Shared ReadOnly Property DefaultCredentials As ICredentials

[JScript] public static function get DefaultCredentials() : ICredentials;

Description

Gets the system credentials of the application.

The **System.Net.CredentialCache.DefaultCredentials** property applies only to NTLM, negotiate, and Kerberos-based authentication.

Add

[C#] public void Add(Uri uriPrefix, string authType, NetworkCredential cred);

[C++] public: void Add(Uri* uriPrefix, String* authType, NetworkCredential*

cred);

[VB] Public Sub Add(ByVal uriPrefix As Uri, ByVal authType As String, ByVal cred As NetworkCredential)

[JScript] public function Add(uriPrefix : Uri, authType : String, cred : NetworkCredential);

Description

Adds a **System.Net.NetworkCredential** instance to the credential cache.

The

System.Net.CredentialCache.Add(System.Uri, System.String, System.Net.NetworkCredential) method places a **System.Net.NetworkCredential** instance into the **System.Net.CredentialCache**. The cache stores credentials in the order in which they are added to it. When the **System.Net.CredentialCache.GetCredential(System.Uri, System.String)** method is called, it returns the proper matching **System.Net.NetworkCredential** instance. A **System.Uri** that specifies the URI prefix of the resources that the credential grants access to. The authentication scheme used by the resource named in *uriPrefix*. The **System.Net.NetworkCredential** to add to the credential cache.

GetCredential

[C#] public NetworkCredential GetCredential(Uri uriPrefix, string authType);

[C++] public: __sealed NetworkCredential* GetCredential(Uri* uriPrefix, String* authType);

[VB] NotOverridable Public Function GetCredential(ByVal uriPrefix As Uri, ByVal authType As String) As NetworkCredential

1 [JScript] public function GetCredential(uriPrefix : Uri, authType : String) :

2 NetworkCredential;

3
4 *Description*

5 Returns the **System.Net.NetworkCredential** instance associated with the
6 specified URI and authentication type.

7 *Return Value:* A **System.Net.NetworkCredential** or, if there is no matching
8 credential in the cache, **null** .

9 The

10 **System.Net.CredentialCache.GetCredential(System.Uri,System.String)**

11 method searches the **System.Net.CredentialCache** and returns the

12 **System.Net.NetworkCredential** instance for the specified URI and authorization
13 type. If the **System.Net.CredentialCache** contains no matching

14 **System.Net.NetworkCredential** instance, **null** is returned. A **System.Uri** that
15 specifies the URI prefix of the resources that the credential grants access to. The
16 authentication scheme used by the resource named in *uriPrefix* .

17 GetEnumerator

18
19 [C#] public IEnumerator GetEnumerator();

20 [C++] public: __sealed IEnumerator* GetEnumerator();

21 [VB] NotOverridable Public Function GetEnumerator() As IEnumerator

22 [JScript] public function GetEnumerator() : IEnumerator;

23
24 *Description*

1 Returns an enumerator that can iterate through the
2 **System.Net.CredentialCache** instance.

3 *Return Value:* An **System.Collections.IEnumerator** for the
4 **System.Net.CredentialCache** .

5 Remove

6
7 [C#] public void Remove(Uri uriPrefix, string authType);

8 [C++] public: void Remove(Uri* uriPrefix, String* authType);

9 [VB] Public Sub Remove(ByVal uriPrefix As Uri, ByVal authType As String)

10 [JScript] public function Remove(uriPrefix : Uri, authType : String);

11
12 *Description*

13 Deletes a **System.Net.NetworkCredential** instance from the cache.

14 The **System.Net.CredentialCache.Remove(System.Uri, System.String)**
15 method removes the specified **System.Net.NetworkCredential** instance from the
16 **System.Net.CredentialCache** . Multiple calls to the
17 **System.Net.CredentialCache.Remove(System.Uri, System.String)** method for
18 the same **System.Net.NetworkCredential** have no effect. A **System.Uri** that
19 specifies the URI prefix of the resources that the credential is used for. The
20 authentication scheme used by the host named in *uriPrefix* .

21 Dns class (System.Net)

22 ToString

23
24
25 *Description*

Provides simple domain name resolution functionality.

The **System.Net.Dns** class is a static class that retrieves information about a specific host from the Internet Domain Name System (DNS).

BeginGetHostByName

```
[C#] public static IAsyncResult BeginGetHostByName(string hostName,
AsyncCallback requestCallback, object stateObject);
[C++] public: static IAsyncResult* BeginGetHostByName(String* hostName,
AsyncCallback* requestCallback, Object* stateObject);
[VB] Public Shared Function BeginGetHostByName(ByVal hostName As String,
ByVal requestCallback As AsyncCallback, ByVal stateObject As Object) As
IAsyncResult
[JScript] public static function BeginGetHostByName(hostName : String,
requestCallback : AsyncCallback, stateObject : Object) : IAsyncResult;
```

Description

Begins an asynchronous request for **System.Net.IPHostEntry** information about the specified DNS host name.

Return Value: An **System.IAsyncResult** instance that references the asynchronous request.

The **System.Net.Dns.BeginGetHostByName(System.String, System.AsyncCallback, System.Object)** method starts an asynchronous request for DNS host information. The asynchronous callback method uses the **System.Net.Dns.EndGetHostByName(System.IAsyncResult)** method to return

the actual host information. A string containing the DNS name of the host. The **System.AsyncCallback**. The State object.

BeginResolve

[C#] public static IAsyncResult BeginResolve(string hostName, AsyncCallback requestCallback, object stateObject);

[C++] public: static IAsyncResult* BeginResolve(String* hostName, AsyncCallback* requestCallback, Object* stateObject);

[VB] Public Shared Function BeginResolve(ByVal hostName As String, ByVal requestCallback As AsyncCallback, ByVal stateObject As Object) As IAsyncResult

[JScript] public static function BeginResolve(hostName : String, requestCallback : AsyncCallback, stateObject : Object) : IAsyncResult;

Description

Begins an asynchronous request to resolve a DNS host name or IP address in dotted-quad notation to an **System.Net.IPAddress** instance.

Return Value: An **System.IAsyncResult** instance that references the asynchronous request.

The

System.Net.Dns.BeginResolve(System.String, System.AsyncCallback, System.Object) method starts an asynchronous request for DNS host information. The asynchronous callback method uses the **System.Net.Dns.EndResolve(System.IAsyncResult)** method to return the actual

1 host information. A string containing the DNS name of the host. The

2 **System.AsyncCallback**. The State object.

3 **EndGetHostByName**

4
5 [C#] public static IPHostEntry EndGetHostByName(IAsyncResult asyncResult);

6 [C++] public: static IPHostEntry* EndGetHostByName(IAsyncResult*

7 asyncResult);

8 [VB] Public Shared Function EndGetHostByName(ByVal asyncResult As

9 IAsyncResult) As IPHostEntry

10 [JScript] public static function EndGetHostByName(asyncResult : IAsyncResult) :

11 IPHostEntry;

12
13 *Description*

14 Ends an asynchronous request for DNS information.

15 *Return Value:* An **System.Net.IPHostEntry** object containin DNS information
16 about a host.

17 The **System.Net.Dns.EndGetHostByName(System.IAsyncResult)**
18 method completes an asynchronous request for DNS information that was started
19 with a call to
20 **System.Net.Dns.BeginGetHostByName(System.String, System.AsyncCallback,**
21 **System.Object)** . The pending request for DNS information.

22 **EndResolve**

23
24 [C#] public static IPHostEntry EndResolve(IAsyncResult asyncResult);

25 [C++] public: static IPHostEntry* EndResolve(IAsyncResult* asyncResult);

```

1 [VB] Public Shared Function EndResolve(ByVal asyncResult As IAsyncResult)
2 As IPHostEntry
3 [JScript] public static function EndResolve(asyncResult : IAsyncResult) :
4 IPHostEntry;

```

Description

Ends an asynchronous request for DNS information.

Return Value: An **System.Net.IPHostEntry** object containin DNS information about a host.

The **System.Net.Dns.EndResolve(System.IAsyncResult)** method completes an asynchronous request for DNS information that was started with a call to **System.Net.Dns.BeginResolve(System.String,System.AsyncCallback,System.Object)** . The pending request for DNS information.

GetHostByAddress

```

17 [C#] public static IPHostEntry GetHostByAddress(IPAddress address);
18 [C++] public: static IPHostEntry* GetHostByAddress(IPAddress* address);
19 [VB] Public Shared Function GetHostByAddress(ByVal address As IPAddress)
20 As IPHostEntry
21 [JScript] public static function GetHostByAddress(address : IPAddress) :
22 IPHostEntry;

```

Description

Creates an **System.Net.IPEndPoint** instance from a specified **System.Net.IPAddress** instance.

Return Value: An **System.Net.IPEndPoint** instance. An **System.Net.IPAddress** instance.

GetHostByAddress

[C#] public static IPHostEntry GetHostByAddress(string address);
[C++] public: static IPHostEntry* GetHostByAddress(String* address);
[VB] Public Shared Function GetHostByAddress(ByVal address As String) As IPHostEntry
[JScript] public static function GetHostByAddress(address : String) : IPHostEntry;

Gets DNS host information for an IP address.

Description

Creates an **System.Net.IPEndPoint** instance from an address in dotted-quad notation ("198.162.1.2").

Return Value: An **System.Net.IPEndPoint** instance. A string that represents an IP address in dotted-quad notation (for example, "192.168.1.2").

GetHostByName

[C#] public static IPHostEntry GetHostByName(string hostName);
[C++] public: static IPHostEntry* GetHostByName(String* hostName);
[VB] Public Shared Function GetHostByName(ByVal hostName As String) As IPHostEntry
[JScript] public static function GetHostByName(hostName : String) :

1 IPHostEntry;

3 *Description*

4 Gets the DNS information for the specified DNS host name.

5 *Return Value:* An **System.Net.IPHostEntry** object containing host information
6 for the address specified in *hostName* .

7 The **System.Net.Dns.GetHostByName(System.String)** method queries
8 the Internet DNS server for host information. A string containing the DNS name
9 of the host.

10 **GetHostName**

12 [C#] public static string GetHostName();

13 [C++] public: static String* GetHostName();

14 [VB] Public Shared Function GetHostName() As String

15 [JScript] public static function GetHostName() : String;

17 *Description*

18 Gets the host name of the local machine.

19 *Return Value:* A string containing the DNS host name of the local machine.

20 **IpToString**

22 [C#] public static string IpToString(int address);

23 [C++] public: static String* IpToString(int address);

24 [VB] Public Shared Function IpToString(ByVal address As Integer) As String

25 [JScript] public static function IpToString(address : int) : String;

Description

Converts an IP address to a dotted-quad string.

Return Value: The string representation of the IP address.

The **System.Net.Dns.IPToString(System.Int32)** method converts an IP address expressed as an integer (for example, 33663168) to an IP address expressed in dotted-quad notation (for example, "192.168.1.2"). The IP address to convert.

Resolve

[C#] public static IPHostEntry Resolve(string hostName);

[C++] public: static IPHostEntry* Resolve(String* hostName);

[VB] Public Shared Function Resolve(ByVal hostName As String) As

IPHostEntry

[JScript] public static function Resolve(hostName : String) : IPHostEntry;

Description

Resolves a DNS host name or IP address in dotted-quad notation to an **System.Net.IPHostEntry** instance.

Return Value: An **System.Net.IPHostEntry** instance containing address information about the host specified in *hostName* .

The **System.Net.Dns.Resolve(System.String)** method queries a DNS server for the IP address associated with a host name or IP address in dotted-quad notation. A DNS-style host name or IP address in dotted-quad notation. (for example, "www.contoso.com " or "192.168.1.2").

DnsPermission class (System.Net)

ToString

Description

Controls rights to access Domain Name System (DNS) servers on the network.

The default allows all local and Intranet zone applications to access DNS services, and no DNS permission for Internet zone applications.

DnsPermission

Example Syntax:

ToString

[C#] public DnsPermission(PermissionState state);

[C++] public: DnsPermission(PermissionState state);

[VB] Public Sub New(ByVal state As PermissionState)

[JScript] public function DnsPermission(state : PermissionState); Creates a new instance of the **System.Net.DnsPermission** class.

Description

Creates a new instance of the **System.Net.DnsPermission** class that either passes all demands or fails all demands.

If *state* is **System.Security.Permissions.PermissionState.Unrestricted** the **System.Net.DnsPermission** instance passes all demands. If *state* contains any

other value, the **System.Net.DnsPermission** instance fails all demands. One of the **System.Security.Permissions.PermissionState** values.

Copy

[C#] public override IPPermission Copy();

[C++] public: IPPermission* Copy();

[VB] Overrides Public Function Copy() As IPPermission

[JScript] public override function Copy() : IPPermission;

Description

Creates an identical copy of the current permission instance.

Return Value: A new instance of the **System.Net.DnsPermission** class that is an identical copy of the current instance.

A copy of a **System.Net.DnsPermission** instance provides the same access to DNS servers as the original permission instance.

FromXml

[C#] public override void FromXml(SecurityElement securityElement);

[C++] public: void FromXml(SecurityElement* securityElement);

[VB] Overrides Public Sub FromXml(ByVal securityElement As SecurityElement)

[JScript] public override function FromXml(securityElement : SecurityElement);

Description

Reconstructs a **System.Net.DnsPermission** instance from an XML encoding.

The **System.Net.DnsPermission.FromXml(System.Security.SecurityElement)** method reconstructs a **System.Net.DnsPermission** instance from an XML encoding defined by **System.Security.SecurityElement** class. The XML encoding to use to reconstruct the **System.Net.DnsPermission** instance.

Intersect

[C#] public override IPPermission Intersect(IPPermission target);

[C++] public: IPPermission* Intersect(IPPermission* target);

[VB] Overrides Public Function Intersect(ByVal target As IPPermission) As

IPPermission

[JScript] public override function Intersect(target : IPPermission) : IPPermission;

Description

Creates a permission instance that is the intersection of the current permission instance and the specified permission instance.

Return Value: A **System.Net.DnsPermission** instance that represents the intersection of the current **System.Net.DnsPermission** instance with the specified **System.Net.DnsPermission** instance, or **null** if the intersection is empty.

The **System.Net.DnsPermission.Intersect(System.Security.IPermission)** method returns a **System.Net.DnsPermission** instance that allows the access defined by both the current **System.Net.DnsPermission** instance and the specified **System.Net.DnsPermission** instance. Any demand must pass both permissions to

pass their intersection. The **System.Net.DnsPermission** instance to combine with the current instance.

IsSubsetOf

[C#] public override bool IsSubsetOf(IPermission target);

[C++] public: bool IsSubsetOf(IPermission* target);

[VB] Overrides Public Function IsSubsetOf(ByVal target As IPermission) As Boolean

[JScript] public override function IsSubsetOf(target : IPermission) : Boolean;

Description

Determines whether the current permission instance is a subset of the specified permission instance.

Return Value: **true** if the current permission instance is a subset of *target* ; otherwise, **false** .

The current **System.Net.DnsPermission** instance is a subset of the specified **System.Net.DnsPermission** instance if the current **System.Net.DnsPermission** instance specifies a set of operations that is wholly contained by the specified **System.Net.DnsPermission** instance. The second **System.Net.DnsPermission** instance to be tested for the subset relationship.

IsUnrestricted

[C#] public bool IsUnrestricted();

[C++] public: __sealed bool IsUnrestricted();

[VB] NotOverridable Public Function IsUnrestricted() As Boolean

1 [JScript] public function IsUnrestricted() : Boolean;

3 *Description*

4 Checks the overall permission state of the object.

5 *Return Value:* **true** if the **System.Net.DnsPermission** instance was created with
6 **System.Security.Permissions.PermissionState.Unrestricted** ; otherwise, **false** .

7 ToXml

9 [C#] public override SecurityElement ToXml();

10 [C++] public: SecurityElement* ToXml();

11 [VB] Overrides Public Function ToXml() As SecurityElement

12 [JScript] public override function ToXml() : SecurityElement;

14 *Description*

15 Creates an XML encoding of a **System.Net.DnsPermission** instance and
16 its current state.

17 *Return Value:* A **System.Security.SecurityElement** instance containing an XML-
18 encoded representation of the security object, including state information.

19 The **System.Net.DnsPermission.ToXml** method creates a
20 **System.Security.SecurityElement** instance to XML-encode a representation of
21 the **System.Net.DnsPermission** instance, including state information.

22 Union

24 [C#] public override IPPermission Union(IPPermission target);

25 [C++] public: IPPermission* Union(IPPermission* target);

1 [VB] Overrides Public Function Union(ByVal target As IPPermission) As

2 IPPermission

3 [JScript] public override function Union(target : IPPermission) : IPPermission;

4
5 *Description*

6 Creates a permission instance that is the union of the current permission
7 instance and the specified permission instance.

8 *Return Value:* A **System.Net.DnsPermission** instance that represents the union of
9 the current **System.Net.DnsPermission** instance with the specified
10 **System.Net.DnsPermission** instance.

11 The **System.Net.DnsPermission.Union(System.Security.IPermission)**
12 method returns a **System.Net.DnsPermission** instance that allows the access
13 defined by either the current **System.Net.DnsPermission** instance and the
14 specified **System.Net.DnsPermission** instance. Any demand that passes either
15 permission passes their union. The **System.Net.DnsPermission** instance to
16 combine with the current instance.

17 DnsPermissionAttribute class (System.Net)

18 Union

19
20
21 *Description*

22 Enables security actions for **System.Net.DnsPermission** to be applied to
23 code using declarative security. This class cannot be inherited.

24 DnsPermissionAttribute

25 *Example Syntax:*

Union

```
[C#] public DnsPermissionAttribute(SecurityAction action);  
[C++] public: DnsPermissionAttribute(SecurityAction action);  
[VB] Public Sub New(ByVal action As SecurityAction)  
[JScript] public function DnsPermissionAttribute(action : SecurityAction);
```

Description

Initializes a new instance of the **System.Net.DnsPermissionAttribute** class with the specified **System.Security.Permissions.SecurityAction** value. One of the **System.Security.Permissions.SecurityAction** values.

Action

TypeId

Unrestricted

CreatePermission

```
[C#] public override IPPermission CreatePermission();  
[C++] public: IPPermission* CreatePermission();  
[VB] Overrides Public Function CreatePermission() As IPPermission  
[JScript] public override function CreatePermission() : IPPermission;
```

Description

Creates and returns a new instance of the **System.Net.DnsPermission** class.

Return Value: An instance of the **System.Net.DnsPermission** class corresponding to the security declaration.

The **System.Net.DnsPermissionAttribute.CreatePermission** method is called by the security system, not by application code.

EndPoint class (System.Net)

ToString

Description

Identifies a network address. This is an **abstract** class.

The **System.Net.EndPoint** class provides an **abstract** base class that represents a network resource or service. Descendant classes combine network connection information to form a connection point to a service.

EndPoint

Example Syntax:

ToString

[C#] protected EndPoint();

[C++] protected: EndPoint();

[VB] Protected Sub New()

[JScript] protected function EndPoint();

AddressFamily

ToString

[C#] public virtual AddressFamily AddressFamily {get;}

```

1 [C++] public: __property virtual AddressFamily get _AddressFamily();
2 [VB] Overridable Public ReadOnly Property AddressFamily As AddressFamily
3 [JScript] public function get AddressFamily() : AddressFamily;

```

Description

Gets the address family to which the endpoint belongs.

The **System.Net.EndPoint.AddressFamily** property specifies the addressing scheme used by the end point's underlying network protocol.

Create

```

11 [C#] public virtual EndPoint Create(SocketAddress socketAddress);
12 [C++] public: virtual EndPoint* Create(SocketAddress* socketAddress);
13 [VB] Overridable Public Function Create(ByVal socketAddress As
14 SocketAddress) As EndPoint
15 [JScript] public function Create(socketAddress : SocketAddress) : EndPoint;

```

Description

Creates an **System.Net.EndPoint** instance from a **System.Net.SocketAddress** instance.

Return Value: A new **System.Net.EndPoint** instance initialized from the specified **System.Net.SocketAddress** instance. The socket address that serves as the endpoint for a connection.

Serialize

```

25 [C#] public virtual SocketAddress Serialize();

```

1 [C++] public: virtual SocketAddress* Serialize();

2 [VB] Overridable Public Function Serialize() As SocketAddress

3 [JScript] public function Serialize() : SocketAddress;

4
5 *Description*

6 Serializes endpoint information into a **System.Net.SocketAddress**
7 instance.

8 *Return Value:* A **System.Net.SocketAddress** instance containing the endpoint
9 information.

10 EndpointPermission class (System.Net)

11 ToString

12
13
14 *Description*

15 Defines an endpoint that is authorized by a **System.Net.SocketPermission**
16 instance.

17 The **System.Net.EndpointPermission** class defines a network endpoint,
18 including host name, network port number, and transport type used to make the
19 connection.

20 Hostname

21 ToString

22
23 [C#] public string Hostname {get;}

24 [C++] public: __property String* get_Hostname();

25 [VB] Public ReadOnly Property Hostname As String

1 [JScript] public function get Hostname() : String;

3 *Description*

4 Gets the DNS host name or IP address of the server associated with this
5 endpoint.

6 Port

7 ToString

9 [C#] public int Port {get;}

10 [C++] public: __property int get_Port();

11 [VB] Public ReadOnly Property Port As Integer

12 [JScript] public function get Port() : int;

14 *Description*

15 Gets the network port number associated with this endpoint.

16 Transport

17 ToString

19 [C#] public TransportType Transport {get;}

20 [C++] public: __property TransportType get_Transport();

21 [VB] Public ReadOnly Property Transport As TransportType

22 [JScript] public function get Transport() : TransportType;

24 *Description*

25 Gets the transport type associated with this endpoint.

Equals

[C#] public override bool Equals(object obj);

[C++] public: bool Equals(Object* obj);

[VB] Overrides Public Function Equals(ByVal obj As Object) As Boolean

[JScript] public override function Equals(obj : Object) : Boolean;

Description

GetHashCode

[C#] public override int GetHashCode();

[C++] public: int GetHashCode();

[VB] Overrides Public Function GetHashCode() As Integer

[JScript] public override function GetHashCode() : int;

Description

ToString

[C#] public override string ToString();

[C++] public: String* ToString();

[VB] Overrides Public Function ToString() As String

[JScript] public override function ToString() : String;

1
2 *Description*

3 Returns a string that represents the current
4 **System.Net.EndpointPermission** instance.

5 *Return Value:* A string that represents the current
6 **System.Net.EndpointPermission** instance.

7 The **System.Net.EndpointPermission.ToString** method returns a string
8 representing the contents for the **System.Net.EndpointPermission** instance. The
9 string is in the form **System.Net.EndpointPermission.Hostname #**
10 **System.Net.EndpointPermission.Port #**
11 **System.Net.EndpointPermission.Transport .**

12 FileWebRequest class (System.Net)

13 ToString

14
15
16 *Description*

17 Provides a file system implementation of the **System.Net.WebRequest**
18 class.

19 The **System.Net.FileWebRequest** class implements the
20 **System.Net.WebRequest** abstract base class for URIs that use the file:// scheme
21 to request local files.

22 FileWebRequest

23 *Example Syntax:*

24 ToString

```

1
2 [C#] protected FileWebRequest(SerializationInfo serializationInfo,
3 StreamingContext streamingContext);
4 [C++] protected: FileWebRequest(SerializationInfo* serializationInfo,
5 StreamingContext streamingContext);
6 [VB] Protected Sub New(ByVal serializationInfo As SerializationInfo, ByVal
7 streamingContext As StreamingContext)
8 [JScript] protected function FileWebRequest(serializationInfo : SerializationInfo,
9 streamingContext : StreamingContext);
10

```

Description

Initializes a new instance of the **System.Net.FileWebRequest** class from the specified instances of the **System.Runtime.Serialization.SerializationInfo** and **System.Runtime.Serialization.StreamingContext** classes.

This constructor implements the **System.Runtime.Serialization.ISerializable** interface for the **System.Net.FileWebRequest** class. A **System.Runtime.Serialization.SerializationInfo** instance that contains the information required to serialize the new **System.Net.FileWebRequest** instance. An instance of the **System.Runtime.Serialization.StreamingContext** class that contains the source of the serialized stream associated with the new **System.Net.FileWebRequest** instance.

ConnectionGroupName

ToString

1 [C#] public override string ConnectionGroupName {get; set;}

2 [C++] public: __property virtual String* get_ConnectionGroupName();public:

3 __property virtual void set_ConnectionGroupName(String*);

4 [VB] Overrides Public Property ConnectionGroupName As String

5 [JScript] public function get ConnectionGroupName() : String;public function set

6 ConnectionGroupName(String);

7 *Description*

8
9 Gets or sets the name of the connection group for the request. This property
10 is reserved for future use.

11
12 The **System.Net.FileWebRequest.ConnectionGroupName** property is
13 currently not used by the **System.Net.FileWebRequest** class.

14 ContentLength

15 ToString

16
17 [C#] public override long ContentLength {get; set;}

18 [C++] public: __property virtual __int64 get_ContentLength();public: __property

19 virtual void set_ContentLength(__int64);

20 [VB] Overrides Public Property ContentLength As Long

21 [JScript] public function get ContentLength() : long;public function set

22 ContentLength(long);

23 *Description*

24
25 Gets or sets the content length of the data being sent.

1 ContentType

2 ToString

3
4 [C#] public override string ContentType {get; set;}

5 [C++] public: __property virtual String* get_ContentType();public: __property
6 virtual void set_ContentType(String*);

7 [VB] Overrides Public Property ContentType As String

8 [JScript] public function get ContentType() : String;public function set
9 ContentType(String);

10
11 *Description*

12 Gets or sets the content type of the data being sent. This property is
13 reserved for future use.

14 The **System.Net.FileWebRequest.ContentType** property contains the
15 media type of the data being sent. This is typically the MIME encoding of the
16 content. The **System.Net.FileWebRequest.ContentType** property is currently not
17 used by the **System.Net.FileWebRequest** class.

18 Credentials

19 ToString

20
21 [C#] public override ICredentials Credentials {get; set;}

22 [C++] public: __property virtual ICredentials* get_Credentials();public:
23 __property virtual void set_Credentials(ICredentials*);

24 [VB] Overrides Public Property Credentials As ICredentials

25 [JScript] public function get Credentials() : ICredentials;public function set

1 Credentials(ICredentials);

2
3 *Description*

4 Gets or sets the credentials associated with this request. This property is
5 reserved for future use.

6 Because the **System.Net.FileWebRequest** class does not authenticate
7 requests for files from the local file system, it ignores the contents, if any, of the
8 **System.Net.FileWebRequest.Credentials** property. Authentication for
9 **System.Net.FileWebRequest** is handled by the access control lists for the file
10 resource in the underlying file system.

11 Headers

12 ToString

13
14 [C#] public override WebHeaderCollection Headers {get;}

15 [C++] public: __property virtual WebHeaderCollection* get_Headers();

16 [VB] Overrides Public ReadOnly Property Headers As WebHeaderCollection

17 [JScript] public function get Headers() : WebHeaderCollection;

18
19 *Description*

20 Gets a collection of the name/value pairs associated with the request. This
21 property is reserved for future use.

22 The **System.Net.FileWebRequest.Headers** property is currently not used
23 by the **System.Net.FileWebRequest** class.

24 Method

25 ToString

1
2 [C#] public override string Method {get; set;}

3 [C++] public: __property virtual String* get_Method();public: __property virtual
4 void set_Method(String*);

5 [VB] Overrides Public Property Method As String

6 [JScript] public function get Method() : String;public function set Method(String);

7
8 *Description*

9 Gets or sets the protocol method used for the request. This property is
10 reserved for future use.

11 The **System.Net.FileWebRequest.Method** property is currently not used
12 by the **System.Net.FileWebRequest** class.

13 PreAuthenticate

14 ToString

15
16 [C#] public override bool PreAuthenticate {get; set;}

17 [C++] public: __property virtual bool get_PreAuthenticate();public: __property
18 virtual void set_PreAuthenticate(bool);

19 [VB] Overrides Public Property PreAuthenticate As Boolean

20 [JScript] public function get PreAuthenticate() : Boolean;public function set
21 PreAuthenticate(Boolean);

22
23 *Description*

24 Gets or sets a value indicating whether to preauthenticate a request. This
25 property is reserved for future use.

1 The **System.Net.FileWebRequest.PreAuthenticate** property is currently
2 not used by the **System.Net.FileWebRequest** class.

3 Proxy

4 ToString

5
6 [C#] public override IWebProxy Proxy {get; set;}

7 [C++] public: __property virtual IWebProxy* get_Proxy();public: __property
8 virtual void set_Proxy(IWebProxy*);

9 [VB] Overrides Public Property Proxy As IWebProxy

10 [JScript] public function get Proxy() : IWebProxy;public function set
11 Proxy(IWebProxy);

12
13 *Description*

14 Gets or sets the network proxy to use for this request. This property is
15 reserved for future use.

16 The **System.Net.FileWebRequest.Proxy** property is currently not used by
17 the **System.Net.FileWebRequest** class.

18 RequestUri

19 ToString

20
21 [C#] public override Uri RequestUri {get;}

22 [C++] public: __property virtual Uri* get_RequestUri();

23 [VB] Overrides Public ReadOnly Property RequestUri As Uri

24 [JScript] public function get RequestUri() : Uri;

Description

Gets the URI of the request.

Timeout

ToString

[C#] public override int Timeout {get; set;}

[C++] public: __property virtual int get_Timeout();public: __property virtual void set_Timeout(int);

[VB] Overrides Public Property Timeout As Integer

[JScript] public function get Timeout() : int;public function set Timeout(int);

Description

Gets or sets the length of time until the request times out.

BeginGetRequestStream

[C#] public override IAsyncResult BeginGetRequestStream(AsyncCallback callback, object state);

[C++] public: IAsyncResult* BeginGetRequestStream(AsyncCallback* callback, Object* state);

[VB] Overrides Public Function BeginGetRequestStream(ByVal callback As AsyncCallback, ByVal state As Object) As IAsyncResult

[JScript] public override function BeginGetRequestStream(callback : AsyncCallback, state : Object) : IAsyncResult;

1
2 *Description*

3 Begins an asynchronous request for a **System.IO.Stream** instance to use to
4 write data.

5 *Return Value:* An **System.IAsyncResult** that references the asynchronous request.

6 The
7 **System.Net.FileWebRequest.BeginGetRequestStream(System.AsyncCallback**
8 **,System.Object)** method starts an asynchronous request for a stream used to send
9 data to a file system resource. The callback method that implements the
10 **System.AsyncCallback** delegate uses the
11 **System.Net.FileWebRequest.EndGetRequestStream(System.IAsyncResult)**
12 method to return the request stream. The **System.AsyncCallback** delegate. An
13 object containing state information for this request.

14 **BeginGetResponse**

15
16 [C#] public override IAsyncResult BeginGetResponse(AsyncCallback callback,
17 object state);

18 [C++] public: IAsyncResult* BeginGetResponse(AsyncCallback* callback,
19 Object* state);

20 [VB] Overrides Public Function BeginGetResponse(ByVal callback As
21 AsyncCallback, ByVal state As Object) As IAsyncResult

22 [JScript] public override function BeginGetResponse(callback : AsyncCallback,
23 state : Object) : IAsyncResult;

24
25 *Description*

Begins an asynchronous request for a file system resource.

Return Value: An **System.IAsyncResult** that references the asynchronous request.

The

System.Net.FileWebRequest.BeginGetResponse(System.AsyncCallback, System

m.Object) method starts an asynchronous request for a file system resource. The

asynchronous callback method that implements the **System.AsyncCallback**

delegate uses the

System.Net.FileWebRequest.EndGetResponse(System.IAsyncResult) method

to return the actual **System.Net.FileWebResponse**. The **System.AsyncCallback**

delegate. An object containing state information for this request.

EndGetRequestStream

[C#] public override Stream EndGetRequestStream(IAsyncResult asyncResult);

[C++] public: Stream* EndGetRequestStream(IAsyncResult* asyncResult);

[VB] Overrides Public Function EndGetRequestStream(ByVal asyncResult As

IAsyncResult) As Stream

[JScript] public override function EndGetRequestStream(asyncResult :

IAsyncResult) : Stream;

Description

Ends an asynchronous request for a **System.IO.Stream** instance that the application uses to write data.

Return Value: A **System.IO.Stream** instance that the application uses to write data.

The

System.Net.FileWebRequest.EndGetRequestStream(System.IAsyncResult)
method completes an asynchronous stream request that was started by the
**System.Net.FileWebRequest.BeginGetRequestStream(System.AsyncCallback
,System.Object)** method. An **System.IAsyncResult** referencing the pending
request for a stream.

EndGetResponse

[C#] public override WebResponse EndGetResponse(IAsyncResult asyncResult);
[C++] public: WebResponse* EndGetResponse(IAsyncResult* asyncResult);
[VB] Overrides Public Function EndGetResponse(ByVal asyncResult As
IAsyncResult) As WebResponse
[JScript] public override function EndGetResponse(asyncResult : IAsyncResult) :
WebResponse;

Description

Ends an asynchronous request for a file system resource.

Return Value: A **System.Net.WebResponse** that contains the response from the
file system resource.

The

System.Net.FileWebRequest.EndGetResponse(System.IAsyncResult) method
completes an asynchronous request for a file system resource that was started with
the
System.Net.FileWebRequest.BeginGetResponse(System.AsyncCallback, Syste

1 **m.Object)** method. An **System.IAsyncResult** referencing the pending request for
2 a response.

3 **GetRequestStream**

4
5 [C#] public override Stream GetRequestStream();

6 [C++] public: Stream* GetRequestStream();

7 [VB] Overrides Public Function GetRequestStream() As Stream

8 [JScript] public override function GetRequestStream() : Stream;

9
10 *Description*

11 Returns a **System.IO.Stream** instance for writing data to the file system
12 resource.

13 *Return Value:* A **System.IO.Stream** for writing data to the file system resource.

14 The **System.Net.FileWebRequest.GetRequestStream** method returns a
15 **System.IO.Stream** instance for writing data to the file system resource.

16 **GetResponse**

17
18 [C#] public override WebResponse GetResponse();

19 [C++] public: WebResponse* GetResponse();

20 [VB] Overrides Public Function GetResponse() As WebResponse

21 [JScript] public override function GetResponse() : WebResponse;

22
23 *Description*

1 Returns a response to a file system request.

2 *Return Value:* A **System.Net.WebResponse** that contains the response from the
3 file system resource.

4 The **System.Net.FileWebRequest.GetResponse** method returns a
5 **System.Net.WebResponse** instance containing the response from the file system
6 resource.

7 **ISerializable.GetObjectData**

8
9 [C#] void **ISerializable.GetObjectData**(**SerializationInfo** serializationInfo,
10 **StreamingContext** streamingContext);

11 [C++] void **ISerializable::GetObjectData**(**SerializationInfo*** serializationInfo,
12 **StreamingContext** streamingContext);

13 [VB] Sub **GetObjectData**(ByVal serializationInfo As **SerializationInfo**, ByVal
14 streamingContext As **StreamingContext**) Implements **ISerializable.GetObjectData**

15 [JScript] function **ISerializable.GetObjectData**(serializationInfo : **SerializationInfo**,
16 streamingContext : **StreamingContext**);

17 **FileWebResponse** class (**System.Net**)

18 **ToString**

19
20
21 *Description*

22 Provides a file system implementation of the **System.Net.WebResponse**
23 class.

The **System.Net.FileWebResponse** class implements the **System.Net.WebResponse** abstract base class to return file system resources for the **System.Net.FileWebRequest** class.

FileWebResponse

Example Syntax:

ToString

[C#] protected **FileWebResponse**(**SerializationInfo** serializationInfo, **StreamingContext** streamingContext);

[C++] protected: **FileWebResponse**(**SerializationInfo*** serializationInfo, **StreamingContext** streamingContext);

[VB] Protected Sub New(**ByVal** serializationInfo As **SerializationInfo**, **ByVal** streamingContext As **StreamingContext**)

[JScript] protected function **FileWebResponse**(serializationInfo : **SerializationInfo**, streamingContext : **StreamingContext**);

Description

Initializes a new instance of the **System.Net.FileWebResponse** class from the specified instances of the **System.Runtime.Serialization.SerializationInfo** and **System.Runtime.Serialization.StreamingContext** classes.

This constructor implements the **System.Runtime.Serialization.ISerializable** interface for the **System.Net.FileWebResponse** class. A **System.Runtime.Serialization.SerializationInfo** instance that contains the information required to serialize the new **System.Net.FileWebResponse** instance.

1 An instance of the **System.Runtime.Serialization.StreamingContext** class that
2 contains the source of the serialized stream associated with the new
3 **System.Net.FileWebResponse** instance.

4 ContentLength

5 ToString

6
7 [C#] public override long ContentLength {get;}

8 [C++] public: __property virtual __int64 get_ContentLength();

9 [VB] Overrides Public ReadOnly Property ContentLength As Long

10 [JScript] public function get ContentLength() : long;

11
12 *Description*

13 Gets the length of the content in the file system resource.

14 The **System.Net.FileWebResponse.ContentLength** property contains the
15 length, in bytes, of the file system resource.

16 ContentType

17 ToString

18
19 [C#] public override string ContentType {get;}

20 [C++] public: __property virtual String* get_ContentType();

21 [VB] Overrides Public ReadOnly Property ContentType As String

22 [JScript] public function get ContentType() : String;

23
24 *Description*

25 Gets the content type of the file system resource.

1 The **System.Net.FileWebResponse.ContentType** property contains the
2 content type of the file system resource. The value of
3 **System.Net.FileWebResponse.ContentType** is always "binary/octet-stream".

4 Headers

5 ToString

6
7 [C#] public override WebHeaderCollection Headers {get;}

8 [C++] public: __property virtual WebHeaderCollection* get_Headers();

9 [VB] Overrides Public ReadOnly Property Headers As WebHeaderCollection

10 [JScript] public function get Headers() : WebHeaderCollection;

11
12 *Description*

13 Gets a collection of header name/value pairs associated with the response.

14 The **System.Net.FileWebResponse.Headers** property contains two
15 name/value pairs, one for content length and one for content type, both of which
16 are also exposed as properties, **System.Net.FileWebResponse.ContentLength**
17 and **System.Net.FileWebResponse.ContentType** .

18 ResponseUri

19 ToString

20
21 [C#] public override Uri ResponseUri {get;}

22 [C++] public: __property virtual Uri* get_ResponseUri();

23 [VB] Overrides Public ReadOnly Property ResponseUri As Uri

24 [JScript] public function get ResponseUri() : Uri;

Description

Gets the URI of the file system resource that provided the response.

The **System.Net.FileWebResponse.ResponseUri** property contains the URI of the file system resource that provided the response. This is always the file system resource that was requested.

Close

[C#] public override void Close();

[C++] public: void Close();

[VB] Overrides Public Sub Close()

[JScript] public override function Close();

Description

Closes the response stream.

The **System.Net.FileWebResponse.Close** method cleans up the resources used by a **System.Net.FileWebResponse** and closes the response stream by calling the **System.IO.Stream.Close** method.

Dispose

[C#] protected virtual void Dispose(bool disposing);

[C++] protected: virtual void Dispose(bool disposing);

[VB] Overridable Protected Sub Dispose(ByVal disposing As Boolean)

[JScript] protected function Dispose(disposing : Boolean);

GetResponseStream

1
2 [C#] public override Stream GetResponseStream();

3 [C++] public: Stream* GetResponseStream();

4 [VB] Overrides Public Function GetResponseStream() As Stream

5 [JScript] public override function GetResponseStream() : Stream;

6
7 *Description*

8 Returns the data stream from the file system resource.

9 *Return Value:* A **System.IO.Stream** for reading data from the file system
10 resource.

11 The **System.Net.FileWebResponse.GetResponseStream** method returns
12 the data stream from the file system resource.

13 **IDisposable.Dispose**

14
15 [C#] void IDisposable.Dispose();

16 [C++] void IDisposable::Dispose();

17 [VB] Sub Dispose() Implements IDisposable.Dispose

18 [JScript] function IDisposable.Dispose();

19 **ISerializable.GetObjectData**

20
21 [C#] void ISerializable.GetObjectData(SerializationInfo serializationInfo,
22 StreamingContext streamingContext);

23 [C++] void ISerializable::GetObjectData(SerializationInfo* serializationInfo,
24 StreamingContext streamingContext);

25 [VB] Sub GetObjectData(ByVal serializationInfo As SerializationInfo, ByVal

streamingContext As StreamingContext) Implements ISerializable.GetObjectData
[JScript] function ISerializable.GetObjectData(serializationInfo : SerializationInfo,
streamingContext : StreamingContext);

GlobalProxySelection class (System.Net)

ToString

Description

Contains a global default proxy instance for all HTTP requests.

The **System.Net.GlobalProxySelection** stores the proxy settings for the default proxy that **System.Net.WebRequest** instances use to contact Internet sites beyond the local network. The default proxy setting is initialized from the global or application configuration file, and can be overridden for individual requests, or disabled by setting the **System.Net.HttpWebRequest.Proxy** property to the result of the **System.Net.GlobalProxySelection.GetEmptyWebProxy** method.

GlobalProxySelection

Example Syntax:

ToString

[C#] public GlobalProxySelection();

[C++] public: GlobalProxySelection();

[VB] Public Sub New()

[JScript] public function GlobalProxySelection();

Select

ToString

```

1
2 [C#] public static IWebProxy Select {get; set;}
3 [C++] public: __property static IWebProxy* get_Select();public: __property static
4 void set_Select(IWebProxy*);
5 [VB] Public Shared Property Select As IWebProxy
6 [JScript] public static function get Select() : IWebProxy;public static function set
7 Select(IWebProxy);
8

```

Description

Gets or sets the global HTTP proxy.

The **System.Net.GlobalProxySelection.Select** method sets the proxy that all **System.Net.HttpWebRequest** instances use.

GetEmptyWebProxy

```

15 [C#] public static IWebProxy GetEmptyWebProxy();
16 [C++] public: static IWebProxy* GetEmptyWebProxy();
17 [VB] Public Shared Function GetEmptyWebProxy() As IWebProxy
18 [JScript] public static function GetEmptyWebProxy() : IWebProxy;
19

```

Description

Returns an empty proxy instance.

Return Value: An **System.Net.IWebProxy** that contains no information.

The **System.Net.GlobalProxySelection.GetEmptyWebProxy** method returns a blank **System.Net.IWebProxy** instance to indicate that no proxy is used to access an Internet resource.

1 HttpContinueDelegate delegate (System.Net)

2 ToString

3
4
5 *Description*

6 Represents the method that notifies callers when a continue response is
7 received by the client. The numeric value of the HTTP status from the server. The
8 headers returned with the 100-continue response from the server.

9 Use **System.Net.HttpContinueDelegate** to specify the callback method to
10 be called when an HTTP 100-continue response is received from the server. When
11 set, the delegate is called whenever protocol responses of type
12 **System.Net.HttpStatusCode.Continue** are received.

13 HttpStatusCode enumeration (System.Net)

14 ToString

15
16
17 *Description*

18 Contains the values of status codes defined for HTTP.

19 The **System.Net.HttpStatusCode** enumeration contains the values of the
20 status codes defined in RFC 2616 for HTTP 1.1.

21 ToString

22
23 [C#] public const HttpStatusCode Accepted;

24 [C++] public: const HttpStatusCode Accepted;

25 [VB] Public Const Accepted As HttpStatusCode

1 [JScript] public var Accepted : HttpStatusCode;

3 *Description*

4 Equivalent to HTTP status 202.

5 ToString

7 [C#] public const HttpStatusCode Ambiguous;

8 [C++] public: const HttpStatusCode Ambiguous;

9 [VB] Public Const Ambiguous As HttpStatusCode

10 [JScript] public var Ambiguous : HttpStatusCode;

12 *Description*

13 Equivalent to HTTP status 300.

14 ToString

16 [C#] public const HttpStatusCode BadGateway;

17 [C++] public: const HttpStatusCode BadGateway;

18 [VB] Public Const BadGateway As HttpStatusCode

19 [JScript] public var BadGateway : HttpStatusCode;

21 *Description*

22 Equivalent to HTTP status 502.

23 ToString

25 [C#] public const HttpStatusCode BadRequest;

1 [C++] public: const HttpStatusCode BadRequest;
2 [VB] Public Const BadRequest As HttpStatusCode
3 [JScript] public var BadRequest : HttpStatusCode;

4
5 *Description*

6 Equivalent to HTTP status 400.

7 ToString

8
9 [C#] public const HttpStatusCode Conflict;
10 [C++] public: const HttpStatusCode Conflict;
11 [VB] Public Const Conflict As HttpStatusCode
12 [JScript] public var Conflict : HttpStatusCode;

13
14 *Description*

15 Equivalent to HTTP status 409.

16 ToString

17
18 [C#] public const HttpStatusCode Continue;
19 [C++] public: const HttpStatusCode Continue;
20 [VB] Public Const Continue As HttpStatusCode
21 [JScript] public var Continue : HttpStatusCode;

22
23 *Description*

24 Equivalent to HTTP status 100.

25 ToString

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

[C#] public const HttpStatusCode Created;
[C++] public: const HttpStatusCode Created;
[VB] Public Const Created As HttpStatusCode
[JScript] public var Created : HttpStatusCode;

Description

Equivalent to HTTP status 201.

ToString

[C#] public const HttpStatusCode ExpectationFailed;
[C++] public: const HttpStatusCode ExpectationFailed;
[VB] Public Const ExpectationFailed As HttpStatusCode
[JScript] public var ExpectationFailed : HttpStatusCode;

Description

Equivalent to HTTP status 417.

ToString

[C#] public const HttpStatusCode Forbidden;
[C++] public: const HttpStatusCode Forbidden;
[VB] Public Const Forbidden As HttpStatusCode
[JScript] public var Forbidden : HttpStatusCode;

Description

Equivalent to HTTP status 403.

ToString

[C#] public const HttpStatusCode Found;

[C++] public: const HttpStatusCode Found;

[VB] Public Const Found As HttpStatusCode

[JScript] public var Found : HttpStatusCode;

Description

Equivalent to HTTP status 302.

ToString

[C#] public const HttpStatusCode GatewayTimeout;

[C++] public: const HttpStatusCode GatewayTimeout;

[VB] Public Const GatewayTimeout As HttpStatusCode

[JScript] public var GatewayTimeout : HttpStatusCode;

Description

Equivalent to HTTP status 504.

ToString

[C#] public const HttpStatusCode Gone;

[C++] public: const HttpStatusCode Gone;

[VB] Public Const Gone As HttpStatusCode

[JScript] public var Gone : HttpStatusCode;

1
2 *Description*

3 Equivalent to HTTP status 410.

4 ToString

5
6 [C#] public const HttpStatusCode HttpVersionNotSupported;

7 [C++] public: const HttpStatusCode HttpVersionNotSupported;

8 [VB] Public Const HttpVersionNotSupported As HttpStatusCode

9 [JScript] public var HttpVersionNotSupported : HttpStatusCode;

10
11 *Description*

12 Equivalent to HTTP status 505.

13 ToString

14
15 [C#] public const HttpStatusCode InternalServerError;

16 [C++] public: const HttpStatusCode InternalServerError;

17 [VB] Public Const InternalServerError As HttpStatusCode

18 [JScript] public var InternalServerError : HttpStatusCode;

19
20 *Description*

21 Equivalent to HTTP status 500.

22 ToString

23
24 [C#] public const HttpStatusCode LengthRequired;

25 [C++] public: const HttpStatusCode LengthRequired;

1 [VB] Public Const LengthRequired As HttpStatusCode

2 [JScript] public var LengthRequired : HttpStatusCode;

4 *Description*

5 Equivalent to HTTP status 411.

6 ToString

8 [C#] public const HttpStatusCode MethodNotAllowed;

9 [C++] public: const HttpStatusCode MethodNotAllowed;

10 [VB] Public Const MethodNotAllowed As HttpStatusCode

11 [JScript] public var MethodNotAllowed : HttpStatusCode;

13 *Description*

14 Equivalent to HTTP status 405.

15 ToString

17 [C#] public const HttpStatusCode Moved;

18 [C++] public: const HttpStatusCode Moved;

19 [VB] Public Const Moved As HttpStatusCode

20 [JScript] public var Moved : HttpStatusCode;

22 *Description*

23 Equivalent to HTTP status 301.

24 ToString

1
2 [C#] public const HttpStatusCode MovedPermanently;
3 [C++] public: const HttpStatusCode MovedPermanently;
4 [VB] Public Const MovedPermanently As HttpStatusCode
5 [JScript] public var MovedPermanently : HttpStatusCode;

6
7 *Description*

8 Equivalent to HTTP status 301.

9 ToString

10
11 [C#] public const HttpStatusCode MultipleChoices;
12 [C++] public: const HttpStatusCode MultipleChoices;
13 [VB] Public Const MultipleChoices As HttpStatusCode
14 [JScript] public var MultipleChoices : HttpStatusCode;

15
16 *Description*

17 Equivalent to HTTP status 300.

18 ToString

19
20 [C#] public const HttpStatusCode NoContent;
21 [C++] public: const HttpStatusCode NoContent;
22 [VB] Public Const NoContent As HttpStatusCode
23 [JScript] public var NoContent : HttpStatusCode;

24
25 *Description*

Equivalent to HTTP status 204.

ToString

[C#] public const HttpStatusCode NonAuthoritativeInformation;

[C++] public: const HttpStatusCode NonAuthoritativeInformation;

[VB] Public Const NonAuthoritativeInformation As HttpStatusCode

[JScript] public var NonAuthoritativeInformation : HttpStatusCode;

Description

Equivalent to HTTP status 203.

ToString

[C#] public const HttpStatusCode NotAcceptable;

[C++] public: const HttpStatusCode NotAcceptable;

[VB] Public Const NotAcceptable As HttpStatusCode

[JScript] public var NotAcceptable : HttpStatusCode;

Description

Equivalent to HTTP status 406.

ToString

[C#] public const HttpStatusCode NotFound;

[C++] public: const HttpStatusCode NotFound;

[VB] Public Const NotFound As HttpStatusCode

[JScript] public var NotFound : HttpStatusCode;

1
2 *Description*

3 Equivalent to HTTP status 404.

4 ToString

5
6 [C#] public const HttpStatusCode NotImplemented;

7 [C++] public: const HttpStatusCode NotImplemented;

8 [VB] Public Const NotImplemented As HttpStatusCode

9 [JScript] public var NotImplemented : HttpStatusCode;

10
11 *Description*

12 Equivalent to HTTP status 501.

13 ToString

14
15 [C#] public const HttpStatusCode NotModified;

16 [C++] public: const HttpStatusCode NotModified;

17 [VB] Public Const NotModified As HttpStatusCode

18 [JScript] public var NotModified : HttpStatusCode;

19
20 *Description*

21 Equivalent to HTTP status 304.

22 ToString

23
24 [C#] public const HttpStatusCode OK;

25 [C++] public: const HttpStatusCode OK;

1 [VB] Public Const OK As HttpStatusCode

2 [JScript] public var OK : HttpStatusCode;

3
4 *Description*

5 Equivalent to HTTP status 200.

6 ToString

7
8 [C#] public const HttpStatusCode PartialContent;

9 [C++] public: const HttpStatusCode PartialContent;

10 [VB] Public Const PartialContent As HttpStatusCode

11 [JScript] public var PartialContent : HttpStatusCode;

12
13 *Description*

14 Equivalent to HTTP status 206.

15 ToString

16
17 [C#] public const HttpStatusCode PaymentRequired;

18 [C++] public: const HttpStatusCode PaymentRequired;

19 [VB] Public Const PaymentRequired As HttpStatusCode

20 [JScript] public var PaymentRequired : HttpStatusCode;

21
22 *Description*

23 Equivalent to HTTP status 402.

24 ToString

25

1
2 [C#] public const HttpStatusCode PreconditionFailed;
3 [C++] public: const HttpStatusCode PreconditionFailed;
4 [VB] Public Const PreconditionFailed As HttpStatusCode
5 [JScript] public var PreconditionFailed : HttpStatusCode;
6

7 *Description*

8 Equivalent to HTTP status 412.

9 ToString
10

11 [C#] public const HttpStatusCode ProxyAuthenticationRequired;
12 [C++] public: const HttpStatusCode ProxyAuthenticationRequired;
13 [VB] Public Const ProxyAuthenticationRequired As HttpStatusCode
14 [JScript] public var ProxyAuthenticationRequired : HttpStatusCode;
15

16 *Description*

17 Equivalent to HTTP status 407.

18 ToString
19

20 [C#] public const HttpStatusCode Redirect;
21 [C++] public: const HttpStatusCode Redirect;
22 [VB] Public Const Redirect As HttpStatusCode
23 [JScript] public var Redirect : HttpStatusCode;
24

25 *Description*

1 Equivalent to HTTP status 302.

2 ToString

3
4 [C#] public const HttpStatusCode RedirectKeepVerb;

5 [C++] public: const HttpStatusCode RedirectKeepVerb;

6 [VB] Public Const RedirectKeepVerb As HttpStatusCode

7 [JScript] public var RedirectKeepVerb : HttpStatusCode;

8
9 *Description*

10 Equivalent to HTTP status 307.

11 ToString

12
13 [C#] public const HttpStatusCode RedirectMethod;

14 [C++] public: const HttpStatusCode RedirectMethod;

15 [VB] Public Const RedirectMethod As HttpStatusCode

16 [JScript] public var RedirectMethod : HttpStatusCode;

17
18 *Description*

19 Equivalent to HTTP status 303.

20 ToString

21
22 [C#] public const HttpStatusCode RequestedRangeNotSatisfiable;

23 [C++] public: const HttpStatusCode RequestedRangeNotSatisfiable;

24 [VB] Public Const RequestedRangeNotSatisfiable As HttpStatusCode

25 [JScript] public var RequestedRangeNotSatisfiable : HttpStatusCode;

1
2 *Description*

3 Equivalent to HTTP status 416.

4 ToString

5
6 [C#] public const HttpStatusCode RequestEntityTooLarge;

7 [C++] public: const HttpStatusCode RequestEntityTooLarge;

8 [VB] Public Const RequestEntityTooLarge As HttpStatusCode

9 [JScript] public var RequestEntityTooLarge : HttpStatusCode;

10
11 *Description*

12 Equivalent to HTTP status 413.

13 ToString

14
15 [C#] public const HttpStatusCode RequestTimeout;

16 [C++] public: const HttpStatusCode RequestTimeout;

17 [VB] Public Const RequestTimeout As HttpStatusCode

18 [JScript] public var RequestTimeout : HttpStatusCode;

19
20 *Description*

21 Equivalent to HTTP status 408.

22 ToString

23
24 [C#] public const HttpStatusCode RequestUriTooLong;

25 [C++] public: const HttpStatusCode RequestUriTooLong;

1 [VB] Public Const RequestUriTooLong As HttpStatusCode

2 [JScript] public var RequestUriTooLong : HttpStatusCode;

3
4 *Description*

5 Equivalent to HTTP status 414.

6 ToString

7
8 [C#] public const HttpStatusCode ResetContent;

9 [C++] public: const HttpStatusCode ResetContent;

10 [VB] Public Const ResetContent As HttpStatusCode

11 [JScript] public var ResetContent : HttpStatusCode;

12
13 *Description*

14 Equivalent to HTTP status 205.

15 ToString

16
17 [C#] public const HttpStatusCode SeeOther;

18 [C++] public: const HttpStatusCode SeeOther;

19 [VB] Public Const SeeOther As HttpStatusCode

20 [JScript] public var SeeOther : HttpStatusCode;

21
22 *Description*

23 Equivalent to HTTP status 303.

24 ToString

1
2 [C#] public const HttpStatusCode ServiceUnavailable;
3 [C++] public: const HttpStatusCode ServiceUnavailable;
4 [VB] Public Const ServiceUnavailable As HttpStatusCode
5 [JScript] public var ServiceUnavailable : HttpStatusCode;

6
7 *Description*

8 Equivalent to HTTP status 503.

9 ToString

10
11 [C#] public const HttpStatusCode SwitchingProtocols;
12 [C++] public: const HttpStatusCode SwitchingProtocols;
13 [VB] Public Const SwitchingProtocols As HttpStatusCode
14 [JScript] public var SwitchingProtocols : HttpStatusCode;

15
16 *Description*

17 Equivalent to HTTP status 101.

18 ToString

19
20 [C#] public const HttpStatusCode TemporaryRedirect;
21 [C++] public: const HttpStatusCode TemporaryRedirect;
22 [VB] Public Const TemporaryRedirect As HttpStatusCode
23 [JScript] public var TemporaryRedirect : HttpStatusCode;

24
25 *Description*

Equivalent to HTTP status 307.

ToString

[C#] public const HttpStatusCode Unauthorized;

[C++] public: const HttpStatusCode Unauthorized;

[VB] Public Const Unauthorized As HttpStatusCode

[JScript] public var Unauthorized : HttpStatusCode;

Description

Equivalent to HTTP status 401.

ToString

[C#] public const HttpStatusCode UnsupportedMediaType;

[C++] public: const HttpStatusCode UnsupportedMediaType;

[VB] Public Const UnsupportedMediaType As HttpStatusCode

[JScript] public var UnsupportedMediaType : HttpStatusCode;

Description

Equivalent to HTTP status 415.

ToString

[C#] public const HttpStatusCode Unused;

[C++] public: const HttpStatusCode Unused;

[VB] Public Const Unused As HttpStatusCode

[JScript] public var Unused : HttpStatusCode;

1
2 *Description*

3 Equivalent to HTTP status 306.

4 ToString

5
6 [C#] public const HttpStatusCode UseProxy;

7 [C++] public: const HttpStatusCode UseProxy;

8 [VB] Public Const UseProxy As HttpStatusCode

9 [JScript] public var UseProxy : HttpStatusCode;

10
11 *Description*

12 Equivalent to HTTP status 305.

13 HttpVersion class (System.Net)

14 ToString

15
16
17 *Description*

18 Defines the HTTP version numbers supported by the

19 **System.Net.HttpWebRequest** and **System.Net.HttpWebResponse** classes.

20 The **System.Net.HttpVersion** class defines the HTTP versions supported
21 by the **System.Net.HttpWebRequest** and **System.Net.HttpWebResponse**
22 classes. The HTTP version number is used to control version-specific features of
23 HTTP, such as pipelining and chunking.

24 ToString

1
2 [C#] public static readonly Version Version10;

3 [C++] public: static Version* Version10;

4 [VB] Public Shared ReadOnly Version10 As Version

5 [JScript] public static var Version10 : Version;

6
7 *Description*

8 Defines a **System.Version** instance for HTTP 1.0.

9 ToString

10
11 [C#] public static readonly Version Version11;

12 [C++] public: static Version* Version11;

13 [VB] Public Shared ReadOnly Version11 As Version

14 [JScript] public static var Version11 : Version;

15
16 *Description*

17 Defines a **System.Version** instance for HTTP 1.1.

18 HttpVersion

19 *Example Syntax:*

20 ToString

21
22 [C#] public HttpVersion();

23 [C++] public: HttpVersion();

24 [VB] Public Sub New()

25 [JScript] public function HttpVersion();

1 HttpRequest class (System.Net)

2 ToString

3
4
5 *Description*

6 Provides an HTTP-specific implementation of the
7 **System.Net.WebRequest** class.

8 The **System.Net.HttpWebRequest** class provides support for the
9 properties and methods defined in **System.Net.WebRequest** and for additional
10 properties and methods that enable the user to interact directly with servers using
11 HTTP.

12 HttpRequest

13 *Example Syntax:*

14 ToString

15
16 [C#] protected HttpRequest(SerializationInfo serializationInfo,
17 StreamingContext streamingContext);

18 [C++] protected: HttpRequest(SerializationInfo* serializationInfo,
19 StreamingContext streamingContext);

20 [VB] Protected Sub New(ByVal serializationInfo As SerializationInfo, ByVal
21 streamingContext As StreamingContext)

22 [JavaScript] protected function HttpRequest(serializationInfo : SerializationInfo,
23 streamingContext : StreamingContext);

24
25 *Description*

1 Initializes a new instance of the **System.Net.HttpWebRequest** class from
2 the specified instances of the **System.Runtime.Serialization.SerializationInfo**
3 and **System.Runtime.Serialization.StreamingContext** classes.

4 This constructor implements the
5 **System.Runtime.Serialization.ISerializable** interface for the
6 **System.Net.HttpWebRequest** class. A
7 **System.Runtime.Serialization.SerializationInfo** instance containing the
8 information required to serialize the new **System.Net.HttpWebRequest** instance.
9 A **System.Runtime.Serialization.StreamingContext** instance containing the
10 source and destination of the serialized stream associated with the new
11 **System.Net.HttpWebRequest** instance.

12 Accept

13 ToString

14
15 [C#] public string Accept {get; set;}

16 [C++] public: __property String* get_Accept();public: __property void
17 set_Accept(String*);

18 [VB] Public Property Accept As String

19 [JScript] public function get Accept() : String;public function set Accept(String);

20
21 *Description*

22 Gets or sets the value of the **Accept** HTTP header.

23 Address

24 ToString

1
2 [C#] public Uri Address {get;}

3 [C++] public: __property Uri* get_Address();

4 [VB] Public ReadOnly Property Address As Uri

5 [JScript] public function get Address() : Uri;

6
7 *Description*

8 Gets the URI of the Internet resource that actually responds to the request.

9 The **System.Net.HttpWebRequest.Address** property is set to the URI that
10 actually responds to a request, after any redirections that might happen during the
11 request are complete.

12 AllowAutoRedirect

13 ToString

14
15 [C#] public bool AllowAutoRedirect {get; set;}

16 [C++] public: __property bool get_AllowAutoRedirect();public: __property void
17 set_AllowAutoRedirect(bool);

18 [VB] Public Property AllowAutoRedirect As Boolean

19 [JScript] public function get AllowAutoRedirect() : Boolean;public function set
20 AllowAutoRedirect(Boolean);

21
22 *Description*

23 Gets or sets a value that indicates whether the request should follow
24 redirection responses.

1 Set **System.Net.HttpWebRequest.AllowAutoRedirect** to **true** if you want
2 the request to automatically follow HTTP redirection headers to the new location
3 of the resource. The maximum number of redirections to follow is set by the
4 **System.Net.HttpWebRequest.MaximumAutomaticRedirections** property.

5 AllowWriteStreamBuffering

6 ToString

7
8 [C#] public bool AllowWriteStreamBuffering {get; set;}

9 [C++] public: __property bool get_AllowWriteStreamBuffering();public:

10 __property void set_AllowWriteStreamBuffering(bool);

11 [VB] Public Property AllowWriteStreamBuffering As Boolean

12 [JScript] public function get AllowWriteStreamBuffering() : Boolean;public

13 function set AllowWriteStreamBuffering(Boolean);

14
15 *Description*

16 Gets or sets a value that indicates whether to buffer the data sent to the
17 Internet resource.

18 When **System.Net.HttpWebRequest.AllowWriteStreamBuffering** is
19 **true** , the data is buffered in memory so it is ready to be resent in the event of
20 redirections or authentication requests.

21 ClientCertificates

22 ToString

23
24 [C#] public X509CertificateCollection ClientCertificates {get;}

25 [C++] public: __property X509CertificateCollection* get_ClientCertificates();

1 [VB] Public ReadOnly Property ClientCertificates As X509CertificateCollection

2 [JScript] public function get ClientCertificates() : X509CertificateCollection;

3
4 *Description*

5 Gets the collection of security certificates associated with this request.

6 Connection

7 ToString

8
9 [C#] public string Connection {get; set;}

10 [C++] public: __property String* get_Connection();public: __property void

11 set_Connection(String*);

12 [VB] Public Property Connection As String

13 [JScript] public function get Connection() : String;public function set

14 Connection(String);

15
16 *Description*

17 Gets or sets the value of the **Connection** HTTP header.

18 The request sends the **System.Net.HttpWebRequest.Connection** property
19 to the Internet resource as the **Connection** HTTP header. If
20 **System.Net.HttpWebRequest.KeepAlive** is true, the value "Keep-alive" is
21 appended to the end of the **Connection** header.

22 ConnectionGroupName

23 ToString

24
25 [C#] public override string ConnectionGroupName {get; set;}

```

1 [C++] public: __property virtual String* get_ConnectionGroupName();public:
2 __property virtual void set_ConnectionGroupName(String*);
3 [VB] Overrides Public Property ConnectionGroupName As String
4 [JScript] public function get ConnectionGroupName() : String;public function set
5 ConnectionGroupName(String);

```

Description

Gets or sets the name of the connection group for the request.

The **System.Net.HttpWebRequest.ConnectionGroupName** property enables you to associate a request with a connection group. This is useful when your application makes requests to one server for different users, such as a Web site that retrieves customer information from a database server.

ContentLength

ToString

```

16 [C#] public override long ContentLength {get; set;}
17 [C++] public: __property virtual __int64 get_ContentLength();public: __property
18 virtual void set_ContentLength(__int64);
19 [VB] Overrides Public Property ContentLength As Long
20 [JScript] public function get ContentLength() : long;public function set
21 ContentLength(long);

```

Description

Gets or sets the **Content-length** HTTP header.

1 The **System.Net.HttpWebRequest.ContentLength** property contains the
2 value to send as the **Content-length** HTTP header with the request.

3 **ContentType**

4 **ToString**

5
6 [C#] public override string ContentType {get; set;}

7 [C++] public: __property virtual String* get_ContentType();public: __property
8 virtual void set_ContentType(String*);

9 [VB] Overrides Public Property ContentType As String

10 [JScript] public function get ContentType() : String;public function set
11 ContentType(String);

12
13 *Description*

14 Gets or sets the value of the **Content-type** HTTP header.

15 The **System.Net.HttpWebRequest.ContentType** property contains the
16 media type of the request. Values assigned to the
17 **System.Net.HttpWebRequest.ContentType** property replace any existing
18 contents when the request sends the **Content-type** HTTP header.

19 **ContinueDelegate**

20 **ToString**

21
22 [C#] public HttpContinueDelegate ContinueDelegate {get; set;}

23 [C++] public: __property HttpContinueDelegate* get_ContinueDelegate();public:
24 __property void set_ContinueDelegate(HttpContinueDelegate*);

25 [VB] Public Property ContinueDelegate As HttpContinueDelegate

1 [JScript] public function get ContinueDelegate() : HttpContinueDelegate;public
2 function set ContinueDelegate(HttpContinueDelegate);

4 *Description*

5 Gets or sets the delegate method called when an HTTP 100-continue
6 response is received from the Internet resource.

7 The **System.Net.HttpWebRequest.ContinueDelegate** property specifies
8 the callback method to call when the client receives a 100-Continue response.

9 CookieContainer

10 ToString

11
12 [C#] public CookieContainer CookieContainer {get; set;}

13 [C++] public: __property CookieContainer* get_CookieContainer();public:

14 __property void set_CookieContainer(CookieContainer*);

15 [VB] Public Property CookieContainer As CookieContainer

16 [JScript] public function get CookieContainer() : CookieContainer;public function
17 set CookieContainer(CookieContainer);

18 19 *Description*

20 Gets or sets the cookies associated with the request.

21 The **System.Net.HttpWebRequest.CookieContainer** property provides an
22 instance of the **System.Net.CookieContainer** class that contains the cookies
23 associated with this request.

24 Credentials

25 ToString

```

1
2 [C#] public override ICredentials Credentials {get; set;}
3 [C++] public: __property virtual ICredentials* get_Credentials();public:
4 __property virtual void set_Credentials(ICredentials*);
5 [VB] Overrides Public Property Credentials As ICredentials
6 [JScript] public function get Credentials() : ICredentials;public function set
7 Credentials(ICredentials);
8

```

9 *Description*

10 Provides authentication information for the request.

11 The **System.Net.HttpWebRequest.Credentials** property contains
12 authentication information to identify the maker of the request. The
13 **System.Net.HttpWebRequest.Credentials** property can be either an instance of
14 **System.Net.NetworkCredential** , in which case the user, password, and domain
15 information contained in the **System.Net.NetworkCredential** instance is used to
16 authenticate the request, or it can be an instance of **System.Net.CredentialCache**
17 , in which case the uniform resource identifier (URI) of the request is used to
18 determine the user, password, and domain information to use to authenticate the
19 request.

20 Expect

21 ToString

```

22
23 [C#] public string Expect {get; set;}
24 [C++] public: __property String* get_Expect();public: __property void
25 set_Expect(String*);

```

1 [VB] Public Property Expect As String

2 [JScript] public function get Expect() : String;public function set Expect(String);

3
4 *Description*

5 Gets or sets the value of the **Expect** HTTP header.

6 By default, **System.Net.HttpWebRequest.Expect** is set to "100-continue".

7 You can add other values to the list that **System.Net.HttpWebRequest.Expect**
8 maintains, or you can delete all values except "100-continue" from the list by
9 setting **System.Net.HttpWebRequest.Expect** to **null** .

10 HaveResponse

11 ToString

12
13 [C#] public bool HaveResponse {get;}

14 [C++] public: __property bool get _HaveResponse();

15 [VB] Public ReadOnly Property HaveResponse As Boolean

16 [JScript] public function get HaveResponse() : Boolean;

17
18 *Description*

19 Gets a value indicating whether a response has been received from an
20 Internet resource.

21 Headers

22 ToString

23
24 [C#] public override WebHeaderCollection Headers {get; set;}

25 [C++] public: __property virtual WebHeaderCollection* get _Headers();public:

1 __property virtual void set_Headers(WebHeaderCollection*);

2 [VB] Overrides Public Property Headers As WebHeaderCollection

3 [JScript] public function get Headers() : WebHeaderCollection;public function set

4 Headers(WebHeaderCollection);

5
6 *Description*

7 Gets a collection of the name/value pairs that make up the HTTP headers.

8 The **System.Net.HttpWebRequest.Headers** collection contains the
9 protocol headers associated with the request. The following table lists the HTTP
10 headers that are not stored in the **System.Net.HttpWebRequest.Headers**
11 collection but are either set by the system or set by properties or methods.

12 IfModifiedSince

13 ToString

14
15 [C#] public DateTime IfModifiedSince {get; set;}

16 [C++] public: __property DateTime get_IfModifiedSince();public: __property
17 void set_IfModifiedSince(DateTime);

18 [VB] Public Property IfModifiedSince As DateTime

19 [JScript] public function get IfModifiedSince() : DateTime;public function set
20 IfModifiedSince(DateTime);

21
22 *Description*

23 Gets or sets the value of the **If-Modified-Since** HTTP header.

24 KeepAlive

25 ToString

[C#] public bool KeepAlive {get; set;}

[C++] public: __property bool get_KeepAlive();public: __property void
set_KeepAlive(bool);

[VB] Public Property KeepAlive As Boolean

[JScript] public function get KeepAlive() : Boolean;public function set
KeepAlive(Boolean);

Description

Gets or sets a value indicating whether to make a persistent connection to the Internet resource.

Set this property to **true** to send an **Connection** HTTP header with the value Keep-alive. An application uses **System.Net.HttpWebRequest.KeepAlive** to indicate a preference for persistent connections. When the **System.Net.HttpWebRequest.KeepAlive** property is **true** , the application makes persistent connections to the servers that support them.

MaximumAutomaticRedirections

ToString

[C#] public int MaximumAutomaticRedirections {get; set;}

[C++] public: __property int get_MaximumAutomaticRedirections();public:
__property void set_MaximumAutomaticRedirections(int);

[VB] Public Property MaximumAutomaticRedirections As Integer

[JScript] public function get MaximumAutomaticRedirections() : int;public
function set MaximumAutomaticRedirections(int);

1
2 *Description*

3 Gets or sets the maximum number of redirects that the request will follow.

4 The **System.Net.HttpWebRequest.MaximumAutomaticRedirections**
5 method property sets the maximum number of redirections for the request to
6 follow if the **System.Net.HttpWebRequest.AllowAutoRedirect** property is **true**

7 .
8 **MediaType**

9 **ToString**

10
11 [C#] public string MediaType {get; set;}

12 [C++] public: __property String* get_MediaType();public: __property void
13 set_MediaType(String*);

14 [VB] Public Property MediaType As String

15 [JScript] public function get MediaType() : String;public function set
16 MediaType(String);

17
18 *Description*

19 Gets or sets the media type of the request.

20 The value of the **System.Net.HttpWebRequest.MediaType** property
21 affects the **System.Net.HttpWebResponse.CharacterSet** property. When you set
22 the **System.Net.HttpWebRequest.MediaType** in the request, the corresponding
23 media type is chosen from the list of character sets returned in the response

24 **Content-type** HTTP header.

25 **Method**

ToString

[C#] public override string Method {get; set;}

[C++] public: __property virtual String* get_Method();public: __property virtual
void set_Method(String*);

[VB] Overrides Public Property Method As String

[JScript] public function get Method() : String;public function set Method(String);

Description

Gets or sets the method for the request.

The **System.Net.HttpWebRequest.Method** property can be set to any of the HTTP 1.1 protocol verbs: GET, HEAD, POST, PUT, DELETE, TRACE, or OPTIONS.

Pipelined

ToString

[C#] public bool Pipelined {get; set;}

[C++] public: __property bool get_Pipelined();public: __property void
set_Pipelined(bool);

[VB] Public Property Pipelined As Boolean

[JScript] public function get Pipelined() : Boolean;public function set
Pipelined(Boolean);

Description

Gets or sets a value indicating whether to pipeline the request to the Internet resource.

An application uses the **System.Net.HttpWebRequest.Pipelined** property to indicate a preference for pipelined connections. When **System.Net.HttpWebRequest.Pipelined** is **true**, an application makes pipelined connections to the servers that support them.

PreAuthenticate

ToString

[C#] public override bool PreAuthenticate {get; set;}

[C++] public: __property virtual bool get_PreAuthenticate();public: __property virtual void set_PreAuthenticate(bool);

[VB] Overrides Public Property PreAuthenticate As Boolean

[JScript] public function get PreAuthenticate() : Boolean;public function set PreAuthenticate(Boolean);

Description

Gets or sets a value indicating whether to send a preauthentication header with the request.

When **System.Net.HttpWebRequest.PreAuthenticate** is **true** and credentials are supplied, the **WWW-authenticate** HTTP header is sent with the initial request if its value is known; otherwise the request uses standard authentication procedures.

ProtocolVersion

ToString

1
2 [C#] public Version ProtocolVersion {get; set;}

3 [C++] public: __property Version* get_ProtocolVersion();public: __property void

4 set_ProtocolVersion(Version*);

5 [VB] Public Property ProtocolVersion As Version

6 [JScript] public function get ProtocolVersion() : Version;public function set

7 ProtocolVersion(Version);

8
9 *Description*

10 Gets or sets the version of HTTP to use for the request.

11 The **System.Net.HttpWebRequest** class supports only versions 1.0 and 1.1
12 of HTTP. Setting **System.Net.HttpWebRequest.ProtocolVersion** to a different
13 version throws an exception.

14 Proxy

15 ToString

16
17 [C#] public override IWebProxy Proxy {get; set;}

18 [C++] public: __property virtual IWebProxy* get_Proxy();public: __property

19 virtual void set_Proxy(IWebProxy*);

20 [VB] Overrides Public Property Proxy As IWebProxy

21 [JScript] public function get Proxy() : IWebProxy;public function set

22 Proxy(IWebProxy);

23
24 *Description*

25 Gets or sets proxy information for the request.

The **System.Net.HttpWebRequest.Proxy** property identifies the **System.Net.WebProxy** instance to use to process requests to Internet resources. To specify that no proxy should be used, set the **System.Net.HttpWebRequest.Proxy** property to the proxy instance returned by the **System.Net.GlobalProxySelection.GetEmptyWebProxy** method.

Referer

ToString

[C#] public string Referer {get; set;}

[C++] public: __property String* get_Referer();public: __property void set_Referer(String*);

[VB] Public Property Referer As String

[JScript] public function get Referer() : String;public function set Referer(String);

Description

Gets or sets the value of the **Referer** HTTP header.

If the **System.Net.HttpWebRequest.AllowAutoRedirect** property is **true**, the **System.Net.HttpWebRequest.Referer** property is set automatically when the request is redirected to another site.

RequestUri

ToString

[C#] public override Uri RequestUri {get;}

[C++] public: __property virtual Uri* get_RequestUri();

[VB] Overrides Public ReadOnly Property RequestUri As Uri

1 [JScript] public function get RequestUri() : Uri;

3 *Description*

4 Gets the original URI of the request.

5 The **System.Uri** instance passed to **System.Net.HttpWebRequest** by the
6 call to **System.Net.WebRequest.Create(System.Uri, System.Boolean)** .

7 SendChunked

8 ToString

10 [C#] public bool SendChunked {get; set;}

11 [C++] public: __property bool get _SendChunked();public: __property void
12 set _SendChunked(bool);

13 [VB] Public Property SendChunked As Boolean

14 [JScript] public function get SendChunked() : Boolean;public function set
15 SendChunked(Boolean);

17 *Description*

18 Gets or sets a value indicating whether to send data in segments to the
19 Internet resource.

20 When **System.Net.HttpWebRequest.SendChunked** is **true** , the request
21 sends data to the Internet resource in segments. The Internet resource must support
22 receiving chunked data.

23 ServicePoint

24 ToString

1
2 [C#] public ServicePoint ServicePoint {get;}

3 [C++] public: __property ServicePoint* get_ServicePoint();

4 [VB] Public ReadOnly Property ServicePoint As ServicePoint

5 [JScript] public function get ServicePoint() : ServicePoint;

6
7 *Description*

8 Gets the service point to use for the request.

9 The value of the **System.Net.HttpWebRequest.ServicePoint** property is
10 **null** until the **System.Net.HttpWebRequest.GetResponse** method is called.

11 Timeout

12 ToString

13
14 [C#] public override int Timeout {get; set;}

15 [C++] public: __property virtual int get_Timeout();public: __property virtual void
16 set_Timeout(int);

17 [VB] Overrides Public Property Timeout As Integer

18 [JScript] public function get Timeout() : int;public function set Timeout(int);

19
20 *Description*

21 Gets or sets the timeout value for a request.

22 **System.Net.HttpWebRequest.Timeout** is the number of milliseconds that
23 a synchronous request made with the
24 **System.Net.HttpWebRequest.GetResponse** method waits for a response. If a
25 resource does not respond within the time-out period, the request throws a

System.Net.WebException with the **System.Net.WebException.Status** property
set to **System.Net.WebExceptionStatus.Timeout** .

TransferEncoding

ToString

[C#] public string TransferEncoding {get; set;}

[C++] public: __property String* get_TransferEncoding();public: __property void
set_TransferEncoding(String*);

[VB] Public Property TransferEncoding As String

[JScript] public function get TransferEncoding() : String;public function set
TransferEncoding(String);

Description

Gets or sets the value of the **Transfer-encoding** HTTP header.

Before you can set the **System.Net.HttpWebRequest.TransferEncoding**
property, you must first set the **System.Net.HttpWebRequest.SendChunked**
property to **true** . Clearing **System.Net.HttpWebRequest.TransferEncoding** by
setting it to **null** has no effect on the value of
System.Net.HttpWebRequest.SendChunked .

UserAgent

ToString

[C#] public string UserAgent {get; set;}

[C++] public: __property String* get_UserAgent();public: __property void
set_UserAgent(String*);

1 [VB] Public Property UserAgent As String

2 [JScript] public function get UserAgent() : String;public function set

3 UserAgent(String);

4
5 *Description*

6 Gets or sets the value of the **User-agent** HTTP header.

7 Abort

8
9 [C#] public override void Abort();

10 [C++] public: void Abort();

11 [VB] Overrides Public Sub Abort()

12 [JScript] public override function Abort();

13
14 *Description*

15 Cancels a request to an Internet resource.

16 **System.Net.HttpWebRequest.Abort** cancels a request to a resource. After
17 a request is canceled, calling **System.Net.HttpWebRequest.GetResponse** ,

18 **System.Net.HttpWebRequest.BeginGetResponse(System.AsyncCallback,Syst**
19 **em.Object)** ,

20 **System.Net.HttpWebRequest.EndGetResponse(System.IAsyncResult)** ,

21 **System.Net.HttpWebRequest.GetRequestStream** ,

22 **System.Net.HttpWebRequest.BeginGetRequestStream(System.AsyncCallbac**
23 **k,System.Object)** , or

24 **System.Net.HttpWebRequest.EndGetRequestStream(System.IAsyncResult)**

will cause a **System.Net.WebException** with **System.Net.WebException.Status** set to **System.Net.WebExceptionStatus.RequestCanceled** .

AddRange

[C#] public void AddRange(int range);

[C++] public: void AddRange(int range);

[VB] Public Sub AddRange(ByVal range As Integer)

[JScript] public function AddRange(range : int);

Description

Adds a byte range header to a request for a specific range from the beginning or end of the requested data.

System.Net.HttpWebRequest.AddRange(System.Int32, System.Int32)

adds a byte range header to the request. The starting or ending point of the range.

AddRange

[C#] public void AddRange(int from, int to);

[C++] public: void AddRange(int from, int to);

[VB] Public Sub AddRange(ByVal from As Integer, ByVal to As Integer)

[JScript] public function AddRange(from : int, to : int); Adds a range header to the request.

Description

Adds a byte range header to the request for a specified range.

System.Net.HttpWebRequest.AddRange(System.Int32, System.Int32)

adds a byte range header to the request. The position at which to start sending data. The position at which to stop sending data.

AddRange

[C#] public void AddRange(string rangeSpecifier, int range);

[C++] public: void AddRange(String* rangeSpecifier, int range);

[VB] Public Sub AddRange(ByVal rangeSpecifier As String, ByVal range As Integer)

[JScript] public function AddRange(rangeSpecifier : String, range : int);

Description

Adds a range header to a request for a specific range from the beginning or end of the requested data.

If *range* is positive, the range is from the start of the data to *range*. The description of the range. The starting or ending point of the range.

AddRange

[C#] public void AddRange(string rangeSpecifier, int from, int to);

[C++] public: void AddRange(String* rangeSpecifier, int from, int to);

[VB] Public Sub AddRange(ByVal rangeSpecifier As String, ByVal from As Integer, ByVal to As Integer)

[JScript] public function AddRange(rangeSpecifier : String, from : int, to : int);

Description

1 Adds a range header to a request for a specified range. The description of
2 the range. The position at which to start sending data. The position at which to
3 stop sending data.

4 BeginGetRequestStream

5
6 [C#] public override IAsyncResult BeginGetRequestStream(AsyncCallback
7 callback, object state);

8 [C++] public: IAsyncResult* BeginGetRequestStream(AsyncCallback* callback,
9 Object* state);

10 [VB] Overrides Public Function BeginGetRequestStream(ByVal callback As
11 AsyncCallback, ByVal state As Object) As IAsyncResult

12 [JScript] public override function BeginGetRequestStream(callback :
13 AsyncCallback, state : Object) : IAsyncResult;

15 Description

16 Begins an asynchronous request for a **System.IO.Stream** instance to use to
17 write data.

18 *Return Value:* An **System.IAsyncResult** that references the asynchronous request.

19 The

20 **System.Net.HttpWebRequest.BeginGetRequestStream(System.AsyncCallbac**
21 **k,System.Object)** method starts an asynchronous request for a stream used to
22 send data for the **System.Net.HttpWebRequest** . The asynchronous callback
23 method uses the

24 **System.Net.HttpWebRequest.EndGetRequestStream(System.IAsyncResult)**

method to return the actual stream. The **System.AsyncCallback** delegate. The state object for this request.

BeginGetResponse

[C#] public override IAsyncResult BeginGetResponse(AsyncCallback callback, object state);

[C++] public: IAsyncResult* BeginGetResponse(AsyncCallback* callback, Object* state);

[VB] Overrides Public Function BeginGetResponse(ByVal callback As AsyncCallback, ByVal state As Object) As IAsyncResult

[JScript] public override function BeginGetResponse(callback : AsyncCallback, state : Object) : IAsyncResult;

Description

Begins an asynchronous request to an Internet resource.

Return Value: An **System.IAsyncResult** that references the asynchronous request for a response.

The

System.Net.HttpWebRequest.BeginGetResponse(System.AsyncCallback, System.Object) method starts an asynchronous request for a response from the Internet resource. The asynchronous callback method uses the **System.Net.HttpWebRequest.EndGetResponse(System.IAsyncResult)** method to return the actual **System.Net.WebResponse**. The **System.AsyncCallback** delegate The state object for this request.

EndGetRequestStream

```

1
2 [C#] public override Stream EndGetRequestStream(IAsyncResult asyncResult);
3 [C++] public: Stream* EndGetRequestStream(IAsyncResult* asyncResult);
4 [VB] Overrides Public Function EndGetRequestStream(ByVal asyncResult As
5 IAsyncResult) As Stream
6 [JScript] public override function EndGetRequestStream(asyncResult :
7 IAsyncResult) : Stream;
8

```

9 *Description*

10 Ends an asynchronous request for a **System.IO.Stream** instance to use to
11 write data.

12 *Return Value:* A **System.IO.Stream** to use to write request data.

13 The
14 **System.Net.HttpWebRequest.EndGetRequestStream(System.IAsyncResult)**
15 method completes an asynchronous request for a stream that was started by the
16 **System.Net.HttpWebRequest.BeginGetRequestStream(System.AsyncCallbac**
17 **k,System.Object)** method. Once the **System.IO.Stream** instance has been
18 returned, you can send data with the **System.Net.HttpWebRequest** by using the
19 **System.IO.Stream.Write(System.Byte[],System.Int32,System.Int32)** method.
20 The pending request for a stream.

21 **EndGetResponse**

```

22
23 [C#] public override WebResponse EndGetResponse(IAsyncResult asyncResult);
24 [C++] public: WebResponse* EndGetResponse(IAsyncResult* asyncResult);
25 [VB] Overrides Public Function EndGetResponse(ByVal asyncResult As

```

1 IAsyncResult) As WebResponse

2 [JScript] public override function EndGetResponse(asyncResult : IAsyncResult) :
3 WebResponse;

4
5 *Description*

6 Ends an asynchronous request to an Internet resource..

7 *Return Value:* A **System.Net.WebResponse** containing the response from the
8 Internet resource.

9 The

10 **System.Net.HttpWebRequest.EndGetResponse(System.IAsyncResult)** method
11 completes an asynchronous request for an Internet resource that was started by
12 calling

13 **System.Net.HttpWebRequest.BeginGetResponse(System.AsyncCallback, Syst**
14 **em.Object)** . The pending request for a response.

15 GetHashCode

16
17 [C#] public override int GetHashCode();

18 [C++] public: int GetHashCode();

19 [VB] Overrides Public Function GetHashCode() As Integer

20 [JScript] public override function GetHashCode() : int;

21
22 *Description*

23 Gets the hash code for this **System.Net.HttpWebRequest** .

24 *Return Value:* The hash code for the **System.Net.HttpWebRequest** .

The hash codes for **System.Net.HttpWebRequest** A and B are guaranteed to be the same when **A.Equals(B)** is **true** .

GetRequestStream

[C#] public override Stream GetRequestStream();

[C++] public: Stream* GetRequestStream();

[VB] Overrides Public Function GetRequestStream() As Stream

[JScript] public override function GetRequestStream() : Stream;

Description

Gets a **System.IO.Stream** instance to use to write request data.

Return Value: A **System.IO.Stream** to use to write request data.

The **System.Net.HttpWebRequest.GetRequestStream** method returns a stream to use to send data for the **System.Net.HttpWebRequest** . Once the **System.IO.Stream** instance has been returned, you can send data with the **System.Net.HttpWebRequest** by using the **System.IO.Stream.Write(System.Byte[],System.Int32,System.Int32)** method.

GetResponse

[C#] public override WebResponse GetResponse();

[C++] public: WebResponse* GetResponse();

[VB] Overrides Public Function GetResponse() As WebResponse

[JScript] public override function GetResponse() : WebResponse;

Description

1 Returns a response from an Internet resource.

2 *Return Value:* A **System.Net.WebResponse** containing the response from the
3 Internet resource.

4 The **System.Net.HttpWebRequest.GetResponse** method returns a
5 **System.Net.WebResponse** instance containing the response from the Internet
6 resource. The actual instance returned is an instance of
7 **System.Net.HttpWebResponse** , and can be typecast to that class to access
8 HTTP-specific properties.

9 SetServicePoint

10
11 [C#] protected void SetServicePoint(ServicePoint servicePoint);

12 [C++] protected: void SetServicePoint(ServicePoint* servicePoint);

13 [VB] Protected Sub SetServicePoint(ByVal servicePoint As ServicePoint)

14 [JScript] protected function SetServicePoint(servicePoint : ServicePoint);

15 ISerializable.GetObjectData

16
17 [C#] void ISerializable.GetObjectData(SerializationInfo serializationInfo,
18 StreamingContext streamingContext);

19 [C++] void ISerializable::GetObjectData(SerializationInfo* serializationInfo,
20 StreamingContext streamingContext);

21 [VB] Sub GetObjectData(ByVal serializationInfo As SerializationInfo, ByVal
22 streamingContext As StreamingContext) Implements ISerializable.GetObjectData

23 [JScript] function ISerializable.GetObjectData(serializationInfo : SerializationInfo,
24 streamingContext : StreamingContext);

25 HttpWebResponse class (System.Net)

ToString

Description

Provides an HTTP-specific implementation of the **System.Net.WebResponse** class.

The **System.Net.HttpWebResponse** class contains support for the properties and methods included in **System.Net.WebResponse** with additional elements that enable the user to interact directly with the HTTP protocol.

HttpWebResponse

Example Syntax:

ToString

[C#] protected HttpWebResponse(SerializationInfo serializationInfo, StreamingContext streamingContext);

[C++] protected: HttpWebResponse(SerializationInfo* serializationInfo, StreamingContext streamingContext);

[VB] Protected Sub New(ByVal serializationInfo As SerializationInfo, ByVal streamingContext As StreamingContext)

[JScript] protected function HttpWebResponse(serializationInfo : SerializationInfo, streamingContext : StreamingContext);

Description

1 Initializes a new instance of the **System.Net.HttpWebResponse** class from
2 the specified **System.Runtime.Serialization.SerializationInfo** and
3 **System.Runtime.Serialization.StreamingContext** instances.

4 This constructor implements the
5 **System.Runtime.Serialization.ISerializable** interface for the
6 **System.Net.HttpWebRequest** class. A
7 **System.Runtime.Serialization.SerializationInfo** containing the information
8 required to serialize the new **System.Net.HttpWebRequest** instance. A
9 **System.Runtime.Serialization.StreamingContext** containing the source of the
10 serialized stream associated with the new **System.Net.HttpWebRequest** instance.

11 CharacterSet

12 ToString

13
14 [C#] public string CharacterSet {get;}

15 [C++] public: __property String* get_CharacterSet();

16 [VB] Public ReadOnly Property CharacterSet As String

17 [JScript] public function get CharacterSet() : String;

18
19 *Description*

20
21 ContentEncoding

22 ToString

23
24 [C#] public string ContentEncoding {get;}

25 [C++] public: __property String* get_ContentEncoding();

1 [VB] Public ReadOnly Property ContentEncoding As String

2 [JScript] public function get ContentEncoding() : String;

3
4 *Description*

5 Gets the method used to encode the body of the response.

6 The **System.Net.HttpWebResponse.ContentEncoding** property contains
7 the value of the **Content-Encoding** header returned with the response.

8 ContentLength

9 ToString

10
11 [C#] public override long ContentLength {get;}

12 [C++] public: __property virtual __int64 get_ContentLength();

13 [VB] Overrides Public ReadOnly Property ContentLength As Long

14 [JScript] public function get ContentLength() : long;

15
16 *Description*

17 Gets the length of the content returned by the request.

18 The **System.Net.HttpWebResponse.ContentLength** property contains the
19 value of the **Content-length** header returned with the response. If the **Content-**
20 **length** header is not set in the response,
21 **System.Net.HttpWebResponse.ContentLength** is set to the value -1.

22 ContentType

23 ToString

24
25 [C#] public override string ContentType {get;}

1 [C++] public: __property virtual String* get_ContentType();

2 [VB] Overrides Public ReadOnly Property ContentType As String

3 [JScript] public function get ContentType() : String;

4
5 *Description*

6 Gets the content type of the response.

7 The **System.Net.HttpWebResponse.ContentType** property contains the
8 value of the **Content-Type** header returned with the response.

9 Cookies

10 ToString

11
12 [C#] public CookieCollection Cookies {get; set;}

13 [C++] public: __property CookieCollection* get_Cookies();public: __property
14 void set_Cookies(CookieCollection*);

15 [VB] Public Property Cookies As CookieCollection

16 [JScript] public function get Cookies() : CookieCollection;public function set
17 Cookies(CookieCollection);

18
19 *Description*

20 Gets or sets the cookies associated with this request.

21 The **System.Net.HttpWebResponse.Cookies** property provides an
22 instance of the **System.Net.CookieCollection** class holding the cookies associated
23 with this response.

24 Headers

25 ToString

```

1
2 [C#] public override WebHeaderCollection Headers {get;}
3 [C++] public: __property virtual WebHeaderCollection* get_Headers();
4 [VB] Overrides Public ReadOnly Property Headers As WebHeaderCollection
5 [JScript] public function get Headers() : WebHeaderCollection;
6

```

Description

Gets the headers associated with this response from the server.

The **System.Net.HttpWebResponse.Headers** property is a collection of name/value pairs containing the HTTP header values returned with the response.

Common headers, listed in the following table, are exposed as properties by the

API: Header Property Content-Encoding

System.Net.HttpWebResponse.ContentEncoding Content-Length

System.Net.HttpWebResponse.ContentLength Content-Type

System.Net.HttpWebResponse.ContentType Last-Modified

System.Net.HttpWebResponse.LastModified Server

System.Net.HttpWebResponse.Server Gets the headers associated with this response from the server.

LastModified

ToString

```

22 [C#] public DateTime LastModified {get;}
23 [C++] public: __property DateTime get_LastModified();
24 [VB] Public ReadOnly Property LastModified As DateTime
25 [JScript] public function get LastModified() : DateTime;

```

1
2 *Description*

3 Gets the last date and time that the contents of the response were modified.

4 The **System.Net.HttpWebResponse.LastModified** property contains the
5 value of the **Last-Modified** header received with the response.

6 Method

7 ToString

8
9 [C#] public string Method {get;}

10 [C++] public: __property String* get_Method();

11 [VB] Public ReadOnly Property Method As String

12 [JScript] public function get Method() : String;

13
14 *Description*

15 Gets the method used to return the response.

16 ProtocolVersion

17 ToString

18
19 [C#] public Version ProtocolVersion {get;}

20 [C++] public: __property Version* get_ProtocolVersion();

21 [VB] Public ReadOnly Property ProtocolVersion As Version

22 [JScript] public function get ProtocolVersion() : Version;

23
24 *Description*

25 Gets the version of the HTTP protocol used in the response.

1 The **System.Net.HttpWebResponse.ProtocolVersion** property contains
2 the HTTP protocol version number of the response sent by the Internet resource.

3 ResponseUri

4 ToString

5
6 [C#] public override Uri ResponseUri {get;}

7 [C++] public: __property virtual Uri* get_ResponseUri();

8 [VB] Overrides Public ReadOnly Property ResponseUri As Uri

9 [JScript] public function get ResponseUri() : Uri;

10
11 *Description*

12 Gets the uniform resource identifier (URI) of the Internet resource that
13 responded to the request. A **System.Uri** instance containing the URI of the Internet
14 resource that responded to the request.

15 The **System.Net.HttpWebResponse.ResponseUri** property contains the
16 URI of the Internet resource that actually responded to the request. This URI may
17 not be the same as the originally requested URI if the request was redirected by
18 the original server.

19 Server

20 ToString

21
22 [C#] public string Server {get;}

23 [C++] public: __property String* get_Server();

24 [VB] Public ReadOnly Property Server As String

25 [JScript] public function get Server() : String;

Description

Gets the name of the server that sent the response.

The **System.Net.HttpWebResponse.Server** property contains the value of the **Server** header returned with the response.

StatusCode

ToString

```
[C#] public HttpStatusCode StatusCode {get;}
```

```
[C++] public: __property HttpStatusCode get_StatusCode();
```

```
[VB] Public ReadOnly Property StatusCode As HttpStatusCode
```

```
[JScript] public function get StatusCode() : HttpStatusCode;
```

Description

Gets the status of the response.

The **System.Net.HttpWebResponse.StatusCode** parameter is a number indicating the status of the HTTP response. The expected values for status are defined in the **System.Net.HttpStatusCode** class.

StatusDescription

ToString

```
[C#] public string StatusDescription {get;}
```

```
[C++] public: __property String* get_StatusDescription();
```

```
[VB] Public ReadOnly Property StatusDescription As String
```

```
[JScript] public function get StatusDescription() : String;
```

1
2 *Description*

3 Gets the status description returned with the response.

4 Close

5
6 [C#] public override void Close();

7 [C++] public: void Close();

8 [VB] Overrides Public Sub Close()

9 [JScript] public override function Close();

10
11 *Description*

12 Closes the response stream.

13 The **System.Net.HttpWebResponse.Close** method closes the response
14 stream and releases the connection to the Internet resource for reuse by other
15 requests.

16 Dispose

17
18 [C#] protected virtual void Dispose(bool disposing);

19 [C++] protected: virtual void Dispose(bool disposing);

20 [VB] Overridable Protected Sub Dispose(ByVal disposing As Boolean)

21 [JScript] protected function Dispose(disposing : Boolean);

22
23 *Description*

24
25 GetHashCode

1
2 [C#] public override int GetHashCode();

3 [C++] public: int GetHashCode();

4 [VB] Overrides Public Function GetHashCode() As Integer

5 [JScript] public override function GetHashCode() : int;

6
7 *Description*

8
9 GetResponseHeader

10
11 [C#] public string GetResponseHeader(string headerName);

12 [C++] public: String* GetResponseHeader(String* headerName);

13 [VB] Public Function GetResponseHeader(ByVal headerName As String) As
14 String

15 [JScript] public function GetResponseHeader(headerName : String) : String;

16
17 *Description*

18 Gets a specified header value returned with the response.

19 *Return Value:* The value of the header specified.

20 The

21 **System.Net.HttpWebResponse.GetResponseHeader(System.String)** method

22 returns the value of the specified header. The header value to return.

23 GetResponseStream

24
25 [C#] public override Stream GetResponseStream();

1 [C++] public: Stream* GetResponseStream();

2 [VB] Overrides Public Function GetResponseStream() As Stream

3 [JScript] public override function GetResponseStream() : Stream;

4
5 *Description*

6 Gets the stream used for reading the body of the response from the server.

7 *Return Value:* A **System.IO.Stream** containing the body of the response.

8 The **System.Net.HttpWebResponse.GetResponseStream** method returns
9 the data stream from the requested Internet resource.

10 IDisposable.Dispose

11
12 [C#] void IDisposable.Dispose();

13 [C++] void IDisposable::Dispose();

14 [VB] Sub Dispose() Implements IDisposable.Dispose

15 [JScript] function IDisposable.Dispose();

16 ISerializable.GetObjectData

17
18 [C#] void ISerializable.GetObjectData(SerializationInfo serializationInfo,

19 StreamingContext streamingContext);

20 [C++] void ISerializable::GetObjectData(SerializationInfo* serializationInfo,

21 StreamingContext streamingContext);

22 [VB] Sub GetObjectData(ByVal serializationInfo As SerializationInfo, ByVal

23 streamingContext As StreamingContext) Implements ISerializable.GetObjectData

24 [JScript] function ISerializable.GetObjectData(serializationInfo : SerializationInfo,

25 streamingContext : StreamingContext);

1 **IAuthenticationModule** interface (System.Net)

2 **ToString**

3
4
5 *Description*

6 Provides the base authentication interface for Web client authentication
7 modules.

8 The **System.Net.IAuthenticationModule** interface defines the properties
9 and methods that custom authentication modules must use.

10 **AuthenticationType**

11 **ToString**

12
13 [C#] string AuthenticationType {get;}

14 [C++] String* get_AuthenticationType();

15 [VB] ReadOnly Property AuthenticationType As String

16 [JScript] abstract function get AuthenticationType() : String;

17
18 *Description*

19 Gets the authentication type provided by this authentication module.

20 The **System.Net.IAuthenticationModule.AuthenticationType** property
21 identifies the authentication type implemented by this authentication module. The
22 **System.Net.IAuthenticationModule.AuthenticationType** property is used by
23 the
24 **System.Net.AuthenticationManager.Register(System.Net.IAuthenticationMo
25 dule)** method to determine if the authentication module has been registered, and

by the

System.Net.AuthenticationManager.Unregister(System.Net.IAuthenticationModule) method to remove a registered authentication module.

CanPreAuthenticate

ToString

[C#] bool CanPreAuthenticate {get;}

[C++] bool get_CanPreAuthenticate();

[VB] ReadOnly Property CanPreAuthenticate As Boolean

[JScript] abstract function get CanPreAuthenticate() : Boolean;

Description

Gets a value indicating whether the authentication module supports preauthentication.

The **System.Net.IAuthenticationModule.CanPreAuthenticate** property is set to **true** to indicate that the authentication module can respond with a valid **System.Net.Authorization** instance when the **System.Net.IAuthenticationModule.PreAuthenticate(System.Net.WebRequest, System.Net.ICredentials)** method is called.

Authenticate

[C#] Authorization Authenticate(string challenge, WebRequest request, ICredentials credentials);

[C++] Authorization* Authenticate(String* challenge, WebRequest* request, ICredentials* credentials);

```

1 [VB] Function Authenticate(ByVal challenge As String, ByVal request As
2 WebRequest, ByVal credentials As ICredentials) As Authorization
3 [JScript] function Authenticate(challenge : String, request : WebRequest,
4 credentials : ICredentials) : Authorization;

```

Description

Returns an instance of the **System.Net.Authorization** class in response to an authentication challenge from a server.

Return Value: An **System.Net.Authorization** instance containing the authorization message for the request, or **null** if the challenge cannot be handled.

The **System.Net.IAuthenticationModule.Authenticate(System.String, System.Net.WebRequest, System.Net.ICredentials)** method conducts the authentication process with the server and returns an **System.Net.Authorization** instance to the **System.Net.AuthenticationManager**. The authentication challenge sent by the server. The **System.Net.WebRequest** instance associated with the challenge. The credentials associated with the challenge.

PreAuthenticate

```

20 [C#] Authorization PreAuthenticate(WebRequest request, ICredentials
21 credentials);
22 [C++] Authorization* PreAuthenticate(WebRequest* request, ICredentials*
23 credentials);
24 [VB] Function PreAuthenticate(ByVal request As WebRequest, ByVal credentials
25 As ICredentials) As Authorization

```